

Scilab Textbook Companion for  
Coulson And Richardson's Chemical  
Engineering, Volume 2  
by J. M. Coulson, J. F. Richardson, J. R.  
Backhurst And J. H. Harker<sup>1</sup>

Created by  
Ankit Mishra  
B.Tech  
Chemical Engineering  
IIT-BHU, VARANASI  
College Teacher  
R.S.singh  
Cross-Checked by

May 20, 2016

<sup>1</sup>Funded by a grant from the National Mission on Education through ICT,  
<http://spoken-tutorial.org/NMEICT-Intro>. This Textbook Companion and Scilab  
codes written in it can be downloaded from the "Textbook Companion Project"  
section at the website <http://scilab.in>

# **Book Description**

**Title:** Coulson And Richardson's Chemical Engineering, Volume 2

**Author:** J. M. Coulson, J. F. Richardson, J. R. Backhurst And J. H. Harker

**Publisher:** Elsevier India

**Edition:** 5

**Year:** 2006

**ISBN:** 9788181471444

Scilab numbering policy used in this document and the relation to the above book.

**Exa** Example (Solved example)

**Eqn** Equation (Particular equation of the above book)

**AP** Appendix to Example(Scilab Code that is an Appednix to a particular Example of the above book)

For example, Exa 3.51 means solved example 3.51 of this book. Sec 2.3 means a scilab code whose theory is explained in Section 2.3 of the book.

# Contents

<b>List of Scilab Codes</b>	<b>4</b>
<b>1 Particulate Solids</b>	<b>5</b>
<b>2 Particle size reduction and enlargement</b>	<b>15</b>
<b>3 Motion of particles in a fluid</b>	<b>19</b>
<b>4 Flow of fluids through granular beds and packed columns</b>	<b>26</b>
<b>5 Sedimentation</b>	<b>28</b>
<b>6 Fluidisation</b>	<b>32</b>
<b>7 Liquid filtration</b>	<b>40</b>
<b>8 Membrane Separation Processes</b>	<b>49</b>
<b>9 Centrifugal Separations</b>	<b>53</b>
<b>10 Leaching</b>	<b>56</b>
<b>11 Distillation</b>	<b>62</b>
<b>12 Absorption of gases</b>	<b>95</b>
<b>13 Liquid liquid extraction</b>	<b>103</b>
<b>14 Evaporation</b>	<b>109</b>

<b>15 Crystallisation</b>	<b>123</b>
<b>16 Drying</b>	<b>131</b>
<b>17 Adsorption</b>	<b>141</b>
<b>18 Ion Exchange</b>	<b>148</b>

# List of Scilab Codes

Exa 1.1	Surface mean diameter . . . . .	5
Exa 1.2	Surface and mass distribution curve and surface mean diameter . . . . .	6
Exa 1.3	variation of mixing index with time . . . . .	8
Exa 1.4	minimum apparent density for separation . . . . .	9
Exa 1.5	efficiency of collection for dust . . . . .	10
Exa 1.6	Overall efficiency of collector . . . . .	11
Exa 1.7	Estimation of particle size . . . . .	13
Exa 2.1	Consumption of energy . . . . .	15
Exa 2.2	Maximum size of the particle . . . . .	16
Exa 2.3	Proposed modifications in ball mill . . . . .	17
Exa 3.1	terminal velocity . . . . .	19
Exa 3.2	Estimation of galena . . . . .	20
Exa 3.3	terminal velocity . . . . .	21
Exa 3.4	Approximate distance travelled . . . . .	23
Exa 3.5	maximum size of crystals . . . . .	24
Exa 4.1	pressure calculation . . . . .	26
Exa 5.1	Minimum area of thickener . . . . .	28
Exa 5.2	Sedimentation velocity and solids flux . . . . .	29
Exa 5.3	Rate of deposition and maximum flux . . . . .	30
Exa 6.1	minimum fluidising velocity . . . . .	32
Exa 6.2	fluidisation and transport of particles . . . . .	33
Exa 6.3	Voidage of the bed . . . . .	34
Exa 6.4	slope of adsorption isotherm . . . . .	36
Exa 6.5	Coefficient of heat transfer between gas and the particles . . . . .	36
Exa 6.6	Volumetric fraction of the bed carrying out evaporation . . . . .	38
Exa 7.1	volume of filtrate collected per cycle . . . . .	40
Exa 7.2	Effect on optimum thickness of the cake . . . . .	42

Exa 7.3	Time taken to produce 1 m <sup>3</sup> of filtrate and pressure in this time . . . . .	43
Exa 7.4	Speed of rotation for maximum throughput . . . . .	44
Exa 7.5	Optimum filtration time for maximum throughput . .	45
Exa 7.6	Thickness of cake produced . . . . .	46
Exa 7.7	Increase in the overall throughput of the press . . . . .	47
Exa 8.2	Area of membrane and average flux . . . . .	49
Exa 8.3	Minimum number of membrane modules required . . .	50
Exa 9.1	Value of capacity factor . . . . .	53
Exa 9.3	TIme taken to produce filtrate . . . . .	54
Exa 10.1	Time required for solute to dissolve . . . . .	56
Exa 10.2	Rate of feed of neutral water to the thickeners . . .	57
Exa 10.3	Required number of thickners . . . . .	58
Exa 10.4	Number of ideal stages required . . . . .	59
Exa 10.5	Number of theoretical stages required . . . . .	60
Exa 11.1	Mole fraction calculation . . . . .	62
Exa 11.2	Saturated Pressure calculation . . . . .	63
Exa 11.3	Vapour phase composition of a mixture . . . . .	64
Exa 11.4	Boiling point of equimolar mixture . . . . .	65
Exa 11.5	Dew point of a equimolar mixture . . . . .	67
Exa 11.6	Composition of vapour and liquid . . . . .	68
Exa 11.7	Number of theoretical plates needed . . . . .	69
Exa 11.8	The Mc Cabe Thiele method . . . . .	71
Exa 11.9	Number of plates required at total reflux . . . . .	73
Exa 11.10	Heat required . . . . .	74
Exa 11.11	Number of theoretical stages required . . . . .	75
Exa 11.12	Amount of distillate . . . . .	76
Exa 11.13	Heat required and average composition . . . . .	77
Exa 11.14	Ideal plates required . . . . .	78
Exa 11.15	Minimum reflux ratio . . . . .	85
Exa 11.16	Minimum reflux ratio . . . . .	89
Exa 11.17	Number of theoretical plates required . . . . .	91
Exa 11.18	Xchange in n with R . . . . .	92
Exa 11.19	Optimum reflux ratio . . . . .	93
Exa 11.20	Plate efficiency for the given data . . . . .	94
Exa 12.1	Overall liquid film coefficient . . . . .	95
Exa 12.2	Mass transfer coefficient . . . . .	96
Exa 12.3	Overall transfer units required . . . . .	98

Exa 12.4	Height of transfer units and number of transfer units . . . . .	99
Exa 12.5	Height of the tower . . . . .	99
Exa 12.6	specific steam consumption . . . . .	101
Exa 13.1	Composition of final raffinate . . . . .	103
Exa 13.3	Overall transfer coefficient . . . . .	105
Exa 13.4	Surface mean droplet size . . . . .	106
Exa 13.5	Number of overall transfer units in raffinate phase . . . . .	107
Exa 14.1	Heat surface required . . . . .	109
Exa 14.2A	Forward feed . . . . .	111
Exa 14.2B	Backward feed . . . . .	115
Exa 14.3	Efficiency of the compressor . . . . .	116
Exa 14.4	Quantity of additional stream required . . . . .	117
Exa 14.5	Capacity and economy of the system . . . . .	118
Exa 14.6	Method to drive the compressor . . . . .	120
Exa 14.7	Optimum boiling time . . . . .	121
Exa 15.1	Supersaturation ratio . . . . .	123
Exa 15.2	Increase in solubility . . . . .	124
Exa 15.3	Theoretical yield . . . . .	125
Exa 15.4	Yield of Sodium acetate . . . . .	125
Exa 15.5	Length of crystalliser . . . . .	127
Exa 15.6	Crystal production rate . . . . .	128
Exa 15.7	Vapour pressure . . . . .	129
Exa 15.8	Mass sublimation rates . . . . .	129
Exa 16.1	Time taken to dry the solids . . . . .	131
Exa 16.2	Time taken to dry the solids . . . . .	132
Exa 16.3	Mass flow rate of dry air . . . . .	133
Exa 16.4	Approximate drying time . . . . .	135
Exa 16.5	Proposed diameter and length . . . . .	136
Exa 16.6	Specified diameter of the bed . . . . .	139
Exa 17.1	Comparison of estimates with the geometric surfaces . . . . .	141
Exa 17.2	Applicability of various equilibrium theories . . . . .	142
Exa 17.3	Length of the bed . . . . .	144
Exa 17.5	Moving bed adsorption design . . . . .	146
Exa 18.1	Prediction of time t against xs values . . . . .	148
Exa 18.2	Concentration of HNO <sub>3</sub> in solution . . . . .	149

# Chapter 1

## Particulate Solids

Scilab code Exa 1.1 Surface mean diameter

```
1 clear;
2 clc;
3 printf("\n Example 1.1");
4 //Given size analysis of a powdered material
5 d=[1,101]; //diameter of the powdered particles
6 x=[0,1]; //mass fractions of the particles
7 plot2d(d,x,style=2,rect=[0,0,120,1])
8 xtitle("size analysis of powder","particle size(um)"
,"mass fraction(x)")
9 d=100*x+1; // from the given plot
10 //calculation of surface mean diameter
11 function[ds]=surface_mean_diameter(x0,x1)
12     ds=1/(integrate('1/(100*x+1)', 'x', x0, x1))
13     funcprot(0)
14 endfunction
15 ds=surface_mean_diameter(0,1); //deduced surface mean
diameter according to def.
16 printf("\n The surface mean diameter is %fum",ds);
```

---

**Scilab code Exa 1.2** Surface and mass distribution curve and surface mean diameter

```
1 clear;
2 clc;
3 printf("\n Example 1.2");
4 //from given differential eq we get these functions
5 //particle number distribution for the size range
6 // 0-10um
7
8 //n=0.5*d^2;
9 //const of integration is0 since at n=0,d=0
10
11 //particle number distribution for the size range
12 // 10-100um
13 //n=83-(0.33*(10^(5))*d^(-3))
14 //c2=83, since at d=10um, n=50
15
16 function[n]= number_distribution(d)
17     if(d<=10) then
18         n=0.5*d^2;
19     else
20         n=83-(0.33*(10^(5))*d^(-3));
21     end
22     funcprot(0)
23 endfunction
24 d=0;
25 while(d<=100)
26     n=number_distribution(d);
27     plot(d,n,"+");
28     d=d+1;
29 end
30 xtitle("number_distribution_plot","diameter (um)",""
31 number distribution");
32 ps=[0 6.2 9.0 10.0 11.4 12.1 13.6 14.7 16.0 17.5
```

```

19.7 22.7 25.5 31.5 100];
32 function[n1]=difference(i)
33 //ps=[0 6.2 9.0 10.0 11.4 12.1 13.6 14.7 16.0 17.5
     19.7 22.7 25.5 31.5 10];
34 // according to the given particle sizes particle
   sizes are in um
35     n1=number_distribution(ps(i+1))-  

         number_distribution(ps(i));
36     funcprot(0);
37 endfunction
38 function[da]=average(i)
39     da= (ps(i+1)+ps(i))/2;
40     funcprot(0);
41 endfunction
42 tot_n1d12=0;
43 tot_n1d13=0;
44 i=1;
45 for i=1:14
46     tot_n1d12=tot_n1d12+difference(i)*(average(i
          ))^2;
47     tot_n1d13=tot_n1d13+difference(i)*(average(i
          ))^3;
48 end
49 printf("\\n tot_n1d12 =%d \\n tot_n1d13=%d",tot_n1d12 ,
      tot_n1d13);
50 function[s]=surface_area(j)
51     s=(difference(j)*(average(j))^2)/tot_n1d12;
52     funcprot(0);
53 endfunction
54 su=0;
55 j=0;
56 xset('window',1);
57
58 plot(0,0,"o-");
59 for j=1:14
60     su=su+surface_area(j);
61     plot(ps(j+1),su,"o-");
62 end

```

```

63 xtitle("surface area and mass distribution plot",
           diameter(um)", "surface area or mass distribution"
           );
64 //mass distribution plot
65 function[x]=mass_distribution(k)
66     x=(difference(k)*(average(k))^3)/tot_n1d13;
67     funcprot(0);
68 endfunction
69 ma=0;
70 k=0;
71 plot(0,0,"+-");
72 for k=1:14
73     ma=ma+mass_distribution(k);
74     plot(ps(k+1),ma,"+-");
75 end
76 //evaluating surface mean diameter
77 function[d]=surface_mean_diameter(l)
78     e=0;
79     for l=1:14
80         n=(mass_distribution(l)/average(l));
81         e=e+n;
82     end
83 d=1/e;
84     funcprot(0);
85 endfunction
86 printf("\nthe surface mean diameter is: %fum",
       surface_mean_diameter());

```

---

### Scilab code Exa 1.3 variation of mixing index with time

```

1 clear all;
2 clc;
3 printf("\n Example 1.3");
4 p=0.20; //components analysed represents 20 percent
          of the mixture by mass

```

```

5 // for a completely unmixed system
6 so=p*(1-p);
7 // for a completely random mixture :
8 n=100; //Each of the sample removed contains 100
    particles
9 sr=p*(1-p)/n;
10 s=[0.025 0.006 0.015 0.018 0.019];
11 time_secs=[30 60 90 120 150];
12 printf("\n degree of mixing is :\n")
13 function[b]=degree_of_mixing()
14 for i=1:5
15     b(i)=(so-s(i))/(so-sr);
16     disp(b(i)); //b is the degree of mixing
17 end
18 return b;
19 funcprot(0)
20 endfunction
21 plot2d(time_secs,degree_of_mixing(),style=3)
22 xtitle("degree of mixing curve","time_secs",""
    degree_of_mixing)
23 //plot of sample variance vs time(secs)
24 xset('window',1)
25 plot2d(time_secs,s,style=2)
26 xtitle("sample variance curve","time_secs","sample
    variance")
27 //from the graph the maxima is at 60 secs

```

---

**Scilab code Exa 1.4** minimum apparent density for separation

```

1 //minimum size of the particle in the mixture of
    quartz and galena(mm)
2 clear all;
3 clc;
4 printf("\n Example 1.4");
5

```

```

6 //maximum size of the particle (mm)
7 d_max=0.065;
8 //minimum size of the particle (mm)
9 d_min=0.015;
10 //density of quartz (kg/m^3)
11 p_quartz=2650;
12 //density of galena (kg/m^3)
13 p_galena=7500;
14 //minimum density of the particle which will give
   this seperation
15 //When stoke's law is applied the required density
   is as given below
16 function[d]=stoke_required_density()
17   p=poly([0], 'p');
18   d=roots((p-7500)-(p-2650)*(d_max/d_min)^2);
19   funcprot(0);
20 endfunction
21 d=stoke_required_density();
22 printf("\n required density is = %d kg/m^3",d);
23 //When Newton's law is applied then the required
   density is as given below
24 function[e]=newton_required_density()
25   r=poly([0], 'r');
26   e=roots((r-7500)-(r-2650)*(d_max/d_min));
27   funcprot(0);
28 endfunction
29 e=newton_required_density();
30 printf("\n required density is by newton law =%d kg/m
   ^3",e);

```

---

### Scilab code Exa 1.5 efficiency of collection for dust

```

1 clear;
2 clc;
3 printf("\n Example 1.5");

```

```

4 //efficiency of the collector for different size
ranges
5 efficiency_1=45; //in percentage for the size range
of 0–5um
6 efficiency_2=80; //in percentage for the size range
of 5–10um
7 efficiency_3=96; //in percentage for the size range
greater than 10um
8
9 //mass percent of the dust for various size range
10 mass_1=50; //in percentage for the size range of 0–5
um
11 mass_2=30; //in percentage for the size range of 5–10
um
12 mass_3=20; //in percentage for the size range
greater than 10um
13 // on the basis of 100kg dust
14 mass_retained_1=0.45*50; //mass retained(kg) in the
size range of 0–5um
15 mass_retained_2=0.80*30; //mass retained(kg) in the
size range of 5–10um
16 mass_retained_3=0.96*20; //mass retained(kg) in the
size range greater than 10um
17 overall_efficiency=0.45*50+0.80*30+0.96*20;
18 printf("\n the overall efficiency is =%f",
overall_efficiency);

```

---

### Scilab code Exa 1.6 Overall efficiency of collector

```

1 clear;
2 clc;
3 printf("\n Example 1.6");
4 //To calculate mass flow of the dust emitted
5 mass_1=10; //in percentage in the size range of 0–5um
6 mass_2=15; //in percentage in the size range of 5–10

```

```

        um
7 mass_3=35; //in percentage in the size range of 10–20
        um
8 mass_4=20; //in percentage in the size range of 20–40
        um
9 mass_5=10; //in percentage in the size range of 40–80
        um
10 mass_6=10; //in percentage in the size range of
        80–160um
11 efficiecnny_1=20; //in percentage in the size range of
        0–5um
12 efficiency_2=40; //in percentage in the size range of
        5–10um
13 efficiency_3=80; //in percentage in the size range of
        10–20um
14 efficiency_4=90; //in percentage in the size range of
        20–40um
15 efficiency_5=95; //in percentage in the size range of
        40–80um
16 efficiency_6=100; //in percentage in the size range
        of 80–160um
17 dust_burden=18; //in g/m^3 at the entrance
18 //taking 1m^3 as the basis of calculation
19 total_mass_retained
        =18*(0.1*0.20+0.15*0.40+0.35*0.80+0.2*0.9+0.1*0.95+0.1*1)
        ;
20 printf("\ntotal mass retained =%fg",
        total_mass_retained);
21 total_efficiency=(total_mass_retained/18)*100;
22 printf("\ntotal efficiency is =%f",total_efficiency)
        ;
23 total_mass_emitted=18-total_mass_retained;
24 printf("total mass emitted is :%fg",
        total_mass_emitted);
25 t=18*(0.1*0.80+0.15*0.60+0.35*0.20);
26 printf("\ntotal mass emitted less than 20um is %fg",
        t);
27 e=t*100/total_mass_emitted;

```

```

28 printf("\nThe efficiency of particles emitted is %f"
29 ,e);
30 //gas flow is 0.3m^3/sec
31 f=0.3*total_mass_emitted;
32 printf("\nmass flow rate is :%fkg/sec",f);

```

---

### Scilab code Exa 1.7 Estimation of particle size

```

1 clear;
2 clc;
3 printf("\n Example 1.7");
4 Ai=(%pi/4)*(0.075)^2; //cross sectional area at the
    gas inlet in m^2
5 do=0.075; //gas outlet diameter in m
6 p=1.3; //gas density in kg/m^3
7 Z=1.2; //height of the separator in m
8 dt=0.3; //separator diameter in m
9 v=1.5; //gas entry velocity in m/sec
10 G=(Ai*v*p); //mass flow rate of the gas in kg/sec
11 printf("\n cross sectional area at the gas inlet is
    %fm^2",Ai);
12 printf("\ngas outlet diameter is %fm",do);
13 printf("\ngas density is %fkg/m^3",p);
14 printf("\nheight of the separator is %fm",Z);
15 printf("\nseparator diameter is%fm",dt);
16 printf("\nmass flow rate of the gas is %fkg/sec",G);
17 function[u]=terminal_vel()
18     u=0.2*(Ai)^2*(do)*p*9.8/(%pi*Z*(dt)*G); //
        velocity is in m/sec
19     funcprot(0);
20 endfunction
21 u=terminal_vel();
22 printf("\nthe terminal velocity of the smallest
        particle retained by the separator =%fm/sec",u);
23 function[d]=particle_diameter(u)

```

```
24     u=terminal_vel();
25     n=0.018*10^(-3); // viscosity in mNs/m^2
26     ps=2700; // density of the particle in kg/m^3
27     d=((u*18*n)/(9.8*(ps-p)))^(0.5); // particle size
          in um
28     funcprot(0);
29 endfunction
30 u=terminal_vel();
31 d=particle_diameter(u);
32 do=d*10^6;
33 printf("\n particle diameter by the stoke law is
          %fum",do);
```

---

# Chapter 2

## Particle size reduction and enlargement

Scilab code Exa 2.1 Consumption of energy

```
1 clear;
2 clc;
3 printf("\n Example 2.1");
4 //Computing energy required in particle size
   reduction by Rittinger's law
5 //energy required in crushing is given by E =Kr.fc
   ((1/L2)-(1/L1))
6 //Given : Energy required in crushing particles from
   50mm to 10mm is           13.0kw/(kg/sec).
7
8 Cr=13.0*(50/4); //Cr = Kr*fc
9
10 //Energy required to crush the particles from 75mm
    to 25mm
11 E = Cr*((1/25)-(1/75));
12 printf("\n The energy required in crushing the
    materials from average particle size of 75mm to
    25mm by Rettingers law is %fkj/kg",E)
13
```

```

14 //Computing the energy required for crushing by Kick
   's law
15 //E = (Kk*fc)*ln(L1/L2) by Kick's law
16
17 Ck = (13.0)/log(50/10); //Ck = Kk*fc
18
19 //Energy required to crush the material from 75mm to
   25mm
20 Ek = Ck*log(75/25);
21 printf("\nThe energy required for crushing the
   material from average particle size of by Kicks
   law 75mm to 25mm is %fkj/kg \n",Ek);
22 printf("\n The size range required is that for
   coarse crushing and Kicks law more closely
   relates the energy required for plastic
   deformation before fracture occurs so the energy
   calculated as that by Kicks law will be taken as
   the more reliable result\n");

```

---

### Scilab code Exa 2.2 Maximum size of the particle

```

1 clear;
2 clc;
3 printf("\n Example 2.2");
4 //Calculating the maximum size of the particle that
   can be fed to the rollers.
5 //Given angle of nip = 31degree
6 //Given diameter of the crushing rolls = 1m
7 //Distance between the crushing rolls is 12.5mm
8 r1 = 0.5; //size of crushing rolls is in meters
9 b = 0.00625; //Distance between the crushing rolls is
   0.0125mm
10 r2 = (r1 + b)/(cos((%pi/180)*15.5))-0.5;
11 printf("\nThe maximum size of the particles which
   should be fed to the rollers : %d mm",r2*10^3);

```

```

12
13 // Calculating the throughput at 2.0 Hz when the
   actual capacity of the machine is 12%.
14 //Working face of the rolls are 0.4m long
15 //bulk density of the feed is 2500kg/m^3
16 printf("\nThe cross sectional area for flow is %.3f
   m^2",0.0125*0.4);
17 printf("\nThe volumetric flow rate is %.2f m^3/sec"
   ,2.0*0.005);
18 printf("\nThe actual throughput is %d kg/sec"
   ,0.010*12*2500/100);

```

---

### Scilab code Exa 2.3 Proposed modifications in ball mill

```

1 clear;
2 clc;
3 printf("\n Example 2.3");
4 //Given diameter of the ball mill is 1.2 meters
5 //Speed of rotation is 0.80Hz
6
7 //for small particles effective radius is 0.6 meters
8 //critical speed of the rotation
9 g=9.80;           //acceleration due to gravity is in m
                     ^2/sec.
10 r=0.6;           //effective radius of rotation is in
                     meters.
11 w = sqrt(g/r);
12 printf("\nThe critical speed of the rotation is %.2f
   rad/sec",w);
13 f=w/(2*pi);    //f is the frequency of the rotation
                     and is in Hz
14 printf("\nThe critical frequency of the rotation is
   %fHz\n",f);
15 optimum_frequency = 0.6*f;
16 printf("The optimum frequency of the rotation is %.2

```

```
f Hz",optimum_frequency);  
17 printf("\nGiven frequency of the rotation is 0.80Hz\n"  
");  
18 printf("The frequency of the rotation should be  
halved\n");  
19 printf("Therefore the optimal frequency is half the  
critical frequency");
```

---

# Chapter 3

## Motion of particles in a fluid

Scilab code Exa 3.1 terminal velocity

```
1 clear all;
2 clc;
3 printf("\n Example 3.1");
4 d = 0.00040;      //Diameter of the particle in m
5 p1 = 820;         //Density of the fluid in kg/m^3
6 meu = 0.01;        //Viscosity of the fluid in N s/m
7 p2 = 7870;        //Density of steel in kg/m^3
8 g = 9.81;          //Acceleartion due to gravity in m
9 //Computation of terminal velocity of a spherical
10 //particle
11 function [x]=galileo_number()
12 x = (2*d^3*(p2-p1)*p1*g)/(3*(meu)^2); //x = (Ro/
13 //pu^2)*Re^2
14 funcprot(0);
15 endfunction
16 x = galileo_number();
17 printf("\n The value of (Ro/pu^2)/Re^2 is %f\n",x);
```

```

18
19 //From table 3.4 log(x) corresponds to log(Re)=0.222
20 Re = 1.667;
21 function[u]=teminal_velocity()
22     u=(Re*meu)/(p1*d);
23     funcprot(0);
24 endfunction
25 u = teminal_velocity();
26 printf("\n The terminal velocity of the steel ball
    is %.3f m/sec or %.0f mm/sec",u,u*1000);

```

---

### Scilab code Exa 3.2 Estimation of galena

```

1 clear;
2 clc;
3 printf("\n Example 3.2");
4 u_water = 5*10^(-3); //The flow velocity of the
    water in m/sec
5 p_galena = 7500; //The density of galena is in
    kg/m^3
6 p_limestone = 2700; //The density of limestone is
    in kg/m^3
7 viscosity = 0.001; //The viscosity of water in N
    s/m^2
8
9 // calculating maximum value of reynold 's number
    considering 5mm particle size
10 Re_max = (u_water*1000*0.0001)/(viscosity);
11 printf("\n The maximum permissible value of Re is %f
    ",Re_max);
12
13 //maximum particle size of galena which will be
    carried away by water
14 d = sqrt((u_water*(18*viscosity))/((7500-1000)*9.81)
    );

```

```

15 printf("\nmaximum particle size of galena which will
           be carried away by water is %.1f um",d*10^(6));
16
17 //maximum particle size of limestone which will be
       carried away by water
18 d1 = sqrt((u_water*(18*viscosity))/((2700-1000)
           *9.81));
19 printf("\nmaximum particle size of limestone which
           will be carried away by water is %.1f um",d1
           *10^(6));
20
21
22 //From the given data 43% galena and 74% limestone
       will be removed .
23 //Given that in the feed there is 20% galena and 80%
       limestone
24 //Assuming 100g feed
25 printf("\n\nIn the overflow:");
26 printf("\nAmount of galena is %fg", (20*0.43));
27 printf("\nAmount of limestone is %fg", (80*0.74));
28 printf("\nconcentration of galena is %.1f per cent"
           ,(20*0.43*100)/(20*0.43+80*0.74));
29 printf("\n\nconcentration of galena is %fper cent"
           ,(80*0.74*100)/(20*0.43+80*0.74));
30 printf("\n\nIn the underflow:")
31 printf("\nconcentration of galena is %.1f percent"
           ,(20*(1-0.43)*100)/(20*(1-0.43)+80*0.26))
32 printf("\nconcentration of limestone is %.1f per
           cent", (80*0.26*100)/(20*0.57+80*0.26))

```

---

### Scilab code Exa 3.3 terminal velocity

```

1 clear;
2 clc;
3 printf("\n Example 3.3");

```

```

4 min_area =6*10^(-6);           //minimum area of mica
    plates in m^2
5 max_area =6*10^(-4);           //maximum area of mica
    plates in m^2
6 p_oil = 820;                  //density of the oil in kg/m
    ^3
7 Viscosity = 0.01;             //Viscosity is in N s/m^2
8 p_mica = 3000;                //Density of mica in kg/m^3
9
10 printf("\n      smallest particles
                    largest particles");
11 printf("\nA:    %fm^2
                    %fm^2
                ",min_area,max_area);
12 printf("\ndp:    %fm
                    %fm
                ",sqrt(4*min_area/(\pi)),sqrt(4*max_area/(\pi)));
13 printf("\ndp^3:%f*10^(-8)m^3
                    %f*10^(-5)m^3
                ",sqrt(4*min_area/(\pi))^3*10^(8),sqrt(4*
                    max_area/(\pi))^3*10^5);
14 printf("\nv:    %f*10^9m^3
                    %f*10^7m^3
                ",0.285*sqrt(4*min_area/(\pi))^3*10^(9),sqrt(4*
                    max_area/(\pi))^3*(10^7)*0.0285);
15 printf("\nk:    %f
                    %f
                ",0.285,0.0285);
16
17 x1 = (4*0.285/(\pi*0.01^2))*(3000-820)
        *(820*2.103*10^(-8)*9.81);
18 x2 = (4*0.0285/(\pi*0.01^2))*(3000-820)
        *(820*2.103*10^(-5)*9.81);
19 printf("\n(Ro/pu^2)Re^2 is %d for the smallest
            particles and %d for the largest particles",x1,x2
        );
20 //From table 3.4 Re for smallest particle is 34.9
        and that for the largest particle is 361

```

```

21 u1 = 34.9*0.01/(820*2.76*10^(-3));
22 printf("\nTerminal velocity for the smallest
           particle is %.3f m/sec",u1);
23 u2 = 361*0.01/(820*2.76*10^(-2));
24 printf("\nTerminal velocity for the largest particle
           is %.3f m/sec",u2);
25 printf("\n\n Thus it is seen that all the particles
           settle at approximately the same velocity");

```

---

### Scilab code Exa 3.4 Approximate distance travelled

```

1 clear;
2 clc;
3 printf("\n Example 3.4");
4 v_particle = 6; //velocity of the particle in m/sec
5 v_water = 1.2; //velocity of the water in m/sec
6 v_rel = v_particle - v_water; //relative velocity of
           particles relative to the fluid in m/sec
7 Re1 = 6*10^(-3)*v_rel*1000/(1*10^(-3)); //Re1 is the
           reynold's no.
8 printf("\nReynold no. is %d",Re1);
9
10 //When the particle has been retarded to a velocity
       such that Re=500
11 ydot = (v_rel*500)/Re1;
12 printf("\nParticle velocity is %.3f m/sec\n",ydot);
13 c = 0.33/(6*10^(-3))*(1000/2500);
14 f = sqrt((3*6*10^(-3)*(2500-1000)*9.81)/1000);
15 function [y]=Fa(t)
16     y = (-1/22)*log(cos(0.517*22*t) + 4.8/0.517*sin
           (0.517*22*t));
17     funcprot(0);
18 endfunction
19
20 function [yd]=deriv(t)

```

```

21      yd= -0.083+(0.517*(9.28*cos(11.37*t) - sin
22          (11.37*t))/(cos(11.37*t) + 9.28*sin(11.37*t))
23          );
24      funcprot(0);
25 endfunction
26
27 function[ydd]=double_deriv(t)
28     ydd = -0.517*(11.37)^2*(9.28*cos(11.37*t) - sin
29         (11.37*t))/(cos(11.37*t) + 9.28*sin(11.37*t))
30         ;
31     funcprot(0);
32 endfunction
33
34 told = 0;
35 while 1
36     tnew = told - deriv(told)/double_deriv(told);
37     if (tnew == told) then
38         y = Fa(told);
39         d = y;
40         printf("\nThe distance moved with speed less
41             than 0.083m/sec is %.3fm",d);
42         t=told;
43         printf("\n The time taken by particle to
44             move this distance is %.3fsec",t);
45         break;
46     end
47     told = tnew;
48 end
49
50 printf("\nThe distance moved by the particle
51     relative to the walls of the plant %.3fm",1.2*t -
52     d);

```

---

### Scilab code Exa 3.5 maximum size of crystals

```
1 clear;
2 clc;
3 printf("\n Example 3.5");
4 // rate of dissolution of salt
5 function[x]=dissolution(d)
6     x = (3*10^(-6))-(2*10^(-4)*3.406*10^(5)*d^2);
7     funcprot(0)
8 endfunction
9
10 // rate of falling of the particle in stokes law
11 // region
12 function[y]=rate_h(d)
13     y = 3.406*d^(2)/(-3*10^(-6)-68.1*d^2); //y is in
14     m/sec
15     funcprot(0);
16 endfunction
17
18 printf("\n By trial and error the solution for d is
19         0.88 mm");
20 printf("\n The rate of dissolution is %f ",
21         dissolution(8.8*10^(-4)));
22 printf("\n The rate of falling of the particle is %f
23         m/sec",rate_h(8.8*10^(-4)));
```

---

# Chapter 4

## Flow of fluids through granular beds and packed columns

Scilab code Exa 4.1 pressure calculation

```
1 clear;
2 clc;
3 printf("\n Example 4.1");
4 //Calculating modified reynold's no.
5 a = 800; //it is in m^2/m^3
6 Product_rate = 0.5; //it is in g/sec
7 Reflux_ratio = 8;
8 Vapour_rate = 4.5; //it is in g/sec
9 G = (4.5*10^(-3))/((%pi/4)*(0.1^2)); // units are in
   kg/m^2.sec
10 meu = 0.02*10^(-3); // units are in Ns/
   m^2
11 e = 0.72;
12
13 Re1 = G/(800*0.28*0.02*10^(-3));
14 printf("\n The modified reynolds no. is %d",Re1);
15
16 x = 4.17/Re1 + 0.29; //x = R1/(pu1^2)
17 printf("\n The value of R1/(pu1^2) is %f",x);
```

```
18
19 l = 16*0.15; //in meters
20 //Solving the integral integral of (pdP) from Pc toPs
21 //=(R1/pu^2)*S*(1-e)*G^2*l/e^3
22 Pc = poly([0], 'Pc');
23 y = roots(151.3-(4.73*10^(-5)*(8000^2-Pc^2)));
24 printf("\n The value of Pc is %dN/m^2",y(1));
```

---

# Chapter 5

## Sedimentation

Scilab code Exa 5.1 Minimum area of thickener

```
1 clear;
2 clc;
3 printf("\n Example 5.1");
4 //Basis 1 kg of solids
5 feed_rate_solid = 1.33; //Mass rate of feed of
    solids in kg/sec
6 U = 1.5; //Mass rate of solids in
    the underflow in kg/sec
7 Y = [5.0 ; 4.2; 3.7 ; 3.1; 2.5];
8 printf("\n concentration(Y) (kg water/kg solids):\n");
9 printf("%.1f\n",Y);
10 printf("\n water to overflow (Y-U) (kg water/kg
    solids):\n");
11 O = Y - 1.5; //Amount of water to overflow in kg
    water/kg solids
12 printf("\n %.1f\n",O);
13 Uc = [2.00*10^(-4) ; 1.20*10^(-4) ; 0.94*10^(-4)
    ; 0.70*10^(-4) ; 0.50*10^(-4)];
14 printf("\n sedimentation rate uc (m/sec):\n");
15 printf("%f \n",Uc);
```

```

16 X
    =[1.75*10^4;2.25*10^4;2.34*10^4;2.29*10^4;2.00*10^4];
    //X = (Y-U)/Uc
17 printf("\n\n(Y-U)/Uc ( s/m) :\n");
18 printf("\n %d\n",X);
19 z = max(X); //prints the maximum value of X
20 printf("\nMaximum value of (Y-U)/Uc = %ds/m" ,z);
21
22 //Calculating the require darea of the thickener
23 A = z*1.33/1000; //1.33 is the mass feed rate of
    solids in kg/sec
24 //1000 is the density of water in kg/m^3
25 printf("\n The required area of the thickener is :%.
    .2 fm^2\n",A);

```

---

### Scilab code Exa 5.2 Sedimentation velocity and solids flux

```

1 clear;
2 clc;
3 printf("\n Example 5.2");
4 //Area of the tank required to give an underflow
    concentration of 1200kg/m^3 for a feed rate of 2
    m^3/min
5
6 //Initial height of slurry in the tank
7 H =
    [900;800;700;600;500;400;300;260;250;220;200;180];
8 uc =
    [13.4;10.76;8.6;6.6;4.9;3.2;1.8;1.21;1.11;0.80;0.60;0.40];
9 i=1;
10 while i<13
11     c(i)=200*900/H(i);
12     x(i)=1000*(1/c(i)-1/1200);

```

```

13     sed(i) = c(i)*uc(i)/(1000*60);
14     y(i)= uc(i)*10^(-3)/((1/c(i)-1/1200)*60);
15     z(i) = 1/y(i);
16     i=i+1;
17 end
18 printf("\nH(mm)");
19 printf("\n%d",H);
20 printf("\n c (kg/m^3):\n");
21 printf("%d\n",c);
22 printf("Sedimentation flux (kg.s/m^2):\n");
23 printf("%.4f\n",x);
24 printf("uc/(1/c-1/1200)\nkg.sec/m^2:\n");
25 printf("%.4f\n",y);
26 printf("1000*(1/c-1/cu)\nm^3/kg*10^3\n");
27 printf("%.3f \n",x);
28 printf("\n\n(1/c-1/1200)/uc\n m^2.kg/sec\n");
29 printf("%.1f\n",z);
30 m1=max
([18.7;20.1;21.3;22.7;23.8;26.0;27.8;30.3;30.0;29.2;27.8;25.0])
;
31 printf("\n\nthe maximum value of (1/c-1/1200)/uc is
%.1f m^2*kg/s",m1);
32 A = 2*200*30.3/60;
33 printf("\n The area required is A = Qc[(1/c-1/cu)/uc
] max = %dm^2",A)

```

---

### Scilab code Exa 5.3 Rate of deposition and maximum flux

```

1 clear;
2 clc;
3 printf("\t Example 5.3 ");
4 //Assumption: Resistance force F on an isolated
    sphere is given by Stoke's law:F = 3*pi(meu)d*u
5
6 C = poly([0], 'C');

```

```

7 x=roots (-4.8*C+(1-C));
8 printf("\n concentration is :%.3f",x);
9
10 //terminal falling velocity u can be calculated by
   force balance
11 // $u = d^2 * g / (18 * \mu_e) * (p_s - p)$ 
12 function[u]=terminal_velocity()
13     d = 10^(-4);      //diameter is in meters
14     g = 9.81;         //acceleration due to gravity is
   in m/sec^2
15     mu_e = 10^(-3); //viscosity is in N.s/m^2
16     ps = 2600;       //density is in kg/m^3
17     p = 1000;        //density is in kg/m^3
18
19     u = (d^2)*g*(ps-p)/(18*mu_e);
20     funcprot(0);
21 endfunction
22
23
24 function[si]=si_max()
25     u=terminal_velocity()
26     printf("\n The terminal falling velocity is %.5f
   m/sec",u);
27     si=u*x*(1-x)^(4.8);
28     funcprot(0);
29 endfunction
30 si = si_max();
31 printf("\nThe maximum value is %f*10^(-4) m^3/m^2 sec
   ",si*10^4)

```

---

# Chapter 6

## Fluidisation

Scilab code Exa 6.1 minimum fluidising velocity

```
1 clear;
2 clc;
3 printf("\n Example 6.1");
4 //Calculating minimum fluidisation velocity
5
6 //Calculating Galileo number
7 function[Ga]=Galileo_number()
8     d = 3*10^(-3);    //particle size is in meters
9     p = 1100;          //density of liquid is in kg/m^3
10    ps = 4200;         //density of spherical particles
11       is in kg/m^3
12    g = 9.81;          //acceleration due to gravity is
13       in m/sec^2
14    u = 3*10^(-3);    //viscosity is in Ns/m^2
15    Ga = d^3*p*(ps-p)*g/u^2;
16    funcprot(0);
17 endfunction
18 Ga = Galileo_number();
19 printf("\nGalileo number = %f*10^5",Ga*10^(-5));
20
21 //Calculating Re mf
```

```

20 Remf = 25.7*(sqrt(1+5.53*10^(-5)*(1.003*10^5))-1);
21 printf("\nValue of Remf is %d",Remf);
22
23 umf = Remf*(3*10^(-3))/(3*10^(-3)*1100);
24 printf("\nminimum fluidisation velocity is %.1f mm/
sec",umf*1000);

```

---

### Scilab code Exa 6.2 fluidisation and transport of particles

```

1 clear;
2 clc;
3 printf("\n Example 6.2");
4 //Calculating voidage by considering eight closely
    packed spheres of diameter d in a cube of size 2d
5 printf("\n (a)");
6 function[e]=voidage()
7     d = 1*10^(-4);      //diameter is in meters
8     meu = 3*10^(-3);    //viscosity is in Ns/m^2
9     ps = 2600;          //density is in kg/m^3
10    p = 900;            //density is in kg/m^3
11    e = [8*d^(3)-8*(pi/6)*d^(3)]/(8*d^(3));
12    funcprot(0);
13 endfunction
14 e = voidage();
15 printf("\nvoidage = %.2f",e);
16
17 //Calculating minimum fluidisation mass flow rate
18
19 function[Gmf]=min_fluidis_vel()
20     e = voidage();
21     d = 1*10^(-4);      //diameter is in meters
22     meu = 3*10^(-3);    //viscosity is in Ns/m^2
23     ps = 2600;          //density is in kg/m^3
24     p = 900;            //density is in kg/m^3
25     g = 9.81;           //acceleration due to gravity

```

```

        is in m/sec^2
26     Gmf = 0.0055*(e)^(3)/(1-e)*(d^2)*p*(ps-p)*g/meu;
27     funcprot(0);
28 endfunction
29 Gmf = min_fluidis_vel();
30 printf("\nminimum fluidisation velocity is %.3f kg/m
^2 sec",Gmf);
31
32
33 printf("\n (b)");
34 function[u]=terminal_velocity()
35     e = voidage();
36     d = 1*10^(-4);      //diameter is in meters
37     meu = 3*10^(-3);   //viscosity is in Ns/m^2
38     ps = 2600;         //density is in kg/m^3
39     p = 900;           //density is in kg/m^3
40     g = 9.81;          //acceleration due to gravity
        is in m/sec^2
41     u = d^2*g*(ps-p)/(18*meu);
42     funcprot(0);
43 endfunction
44 printf("\n terminal velocity is %.4fm/sec",
    terminal_velocity());
45
46 //Reynolds no for this Terminal velocity is
47 Re = (10^(-4)*0.0031*900)/(3*10^(-3));
48 printf("\n Reynolds no =%.3f",Re);
49 printf("\n The required mass flow rate is %.2f kg/m^2
sec",terminal_velocity()*900);

```

---

### Scilab code Exa 6.3 Voidage of the bed

```

1 // to calculate voidage of the bed
2 clear;
3 clc;

```

```

4 printf("\n Example 6.3");
5 function[Gal]=Galileo_number()
6     d = 4*10^(-3); //particle size is in meters
7     p = 1000; //density of water is in kg/m^3
8     ps = 2500; //density of glass is in kg/m^3
9     g = 9.81; //acceleration due to gravity is
               //in m/sec^2
10    u = 1*10^(-3); //viscosity is in Ns/m^2
11    Ga = d^3*p*(ps-p)*g/u^2;
12    funcprot(0);
13 endfunction
14 printf("\nGalileo number = %.2f*10^5",Galileo_number
        ()*10^(-5));
15
16 function[Re]=Reynolds_no()
17     Ga = Galileo_number();
18     Re = (2.33*Ga^(0.018)-1.53*Ga^(-0.016))^(13.3);
19     funcprot(0);
20 endfunction
21 printf("\n The Reynolds no is %d",Reynolds_no());
22 v = Reynolds_no()*(1*10^(-3))/(0.004*1000);
23 printf("\nvelocity = %.2f m/sec",v);
24
25 n = poly([0], 'n');
26 z = roots((4.8-n)-0.043*(Galileo_number())^(0.57)*(n
               -2.4));
27 printf("\nvalue of n is %.2f",z);
28
29 //voidage at a velocity is 0.25m/sec
30 e=0.1;
31 while 1
32     enew = e - ((0.25/0.45)-e^(2.42))/(-2.42*e^1.42);
33     if (enew == e) then
34         printf("\nVoidage is %.3f",e);
35         break;
36     end
37     e=enew;
38 end

```

---

**Scilab code Exa 6.4** slope of adsorption isotherm

```
1 clear;
2 clc;
3 printf("\n Example 6.4");
4 t = 250:250:2000; //time is in secs
5 y =
    [0.00223;0.00601;0.00857;0.0106;0.0121;0.0129;0.0134;0.0137];

6
7 i =1;
8 yo = 0.01442;
9 while i<9
10     z(i) = y(i)/yo;
11     y(i)=1-z(i);
12     x(i)=log(y(i));
13     i=i+1;
14 end
15 xtitle("slope of adsorption isotherm","Time(sec)",""
    log(1-(y/yo)))
16 plot(t,x,"o-")
17 printf("\nFrom the graph the value of slope is %f"
    ,-0.00167);
18 Gm = 0.679*10^(-6); //units are in kmol/sec
19 W = 4.66; //units are in gram
20 b = poly([0], 'b');
21 s = roots(-0.00167*4.66*b+0.679*10^(-6));
22 printf("\n b = %.4f kmol/kg",s*10^3);
```

---

**Scilab code Exa 6.5** Coefficient of heat transfer between gas and the particles

```

1 clear;
2 clc;
3 printf("\n Example 6.5");
4 gas_flow_rate =0.2;           // units are in kg/m^2
5 c = 0.88;                   // specific heat capacity
     of air is kj/kg K
6 viscosity = 0.015*10^(-3); // viscosity is in Ns/m^2
7 d = 0.25*10^(-3);          // particle size is in
     meters
8 k = 0.03;                   // thermal conductivity is
     in W/m K
9 e = 0.57;                   // e is voidage
10
11 T = [339.5;337.7;335.0;333.6;333.3;333.2]; // 
     temperature is in kelvins
12 deltaT = T - 333.2;
13 h = [0;0.64;1.27;1.91;2.54;3.81];
14 xtitle("temperature rise as a function of bed height
     ", "height above bed support(mm)", "deltaT(K)");
15 plot(h,deltaT,'o-');
16
17 //Area under the curve gives the value of the heat
     transfer integral =8.82mm K
18
19 q = 0.2*0.88*(339.5-332.2);
20 printf("\n heat transferred = %.2f kw/m^2 of bed
     cross sectional area",q);
21
22 //Assuming 1m^3 volume
23 Vp = (1-0.57);           //Volume of particles is in m^3
24 printf("\n Volume of particles is %.2f m^3",Vp)
25 v1 = (%pi/6)*d^3;        //Volume of 1 particle in m^3
26 printf("\n Volume of 1 particle is %.2f*10^(-12) m^3
     ",v1*10^(12));
27 printf("\n number of particles is %.2f*10^(10)",Vp/
     v1*10^(-10));
28
29 x =poly([0], 'x');

```

```

30 h = roots(1100 - x*(1.03*10^4)*(8.82*10^(-3)));
31 printf("\n heat transfer coefficient = %.1f W/m^2",h
      );
32
33 //Nu = 0.11Re^(1.28)
34 Re = (0.2*0.25*10^(-3))/(0.015*10^(-3));
35 h1 = 0.11*(Re)^(1.28)*k/d;
36 printf("\n h = %.1f W/m^2 K",h1);

```

---

**Scilab code Exa 6.6** Volumetric fraction of the bed carrying out evaporation

```

1 clear;
2 clc;
3 printf("\n Example 6.6");
4 cp = 0.85; // specific
               heat capacity of the air
5 h = [0 0.625 1.25 1.875 2.5 3.75];
               //height in mm
6 T=[339.5 337.7 335.0 333.6 333.3 333.2]; //
               temperature in K
7 deltaT = T - 333.2; ////
               temperature difference in kelvins
8 plot(h,deltaT,"o-");
9 xtitle("deltaT as a function of bed height","Height
               above bed support z(mm)","Temperature difference
               deltaT (K)");
10 //From the plot area under the curve is 6.31 K mm
11 sp = (6/(0.25*10^(-3)))*(0.5); //sp is surface area
               per unit volume in m^2/m^3
12 G = 0.2; //in kg/m^2 sec
13 Cp = 850; //Cp is in J/kg K
14 h1 = poly([0], 'h1');
15 s = roots(0.2*850*6.3-h1*1.2*10^(4)*6.31*10^(-3));
16 printf("\n Coefficient for heat transfer between

```

```
the gas and the particles= %.1fW/m^2 K" ,s);  
17  
18 printf("\n Let the evaporation rate be 0.1 kg/sec at  
a temp difference = 50 degK");  
19 mdot = 0.1; //evaporation rate is 0.1 kg/sec  
20 Latent_heat = 2.6*10^(6);  
21 printf("\n The heat flow = %.1f*10^5 W" ,mdot*  
Latent_heat*10^(-5));  
22 A=(2.6*10^5)/(14.1*50);  
23 printf("\n the effective area of the bed A = %d m^2"  
,A);  
24 printf("\n The surface area of the bed = %d m^2"  
,0.1*1.2*10^4);  
25 printf("\n hence the fraction of the bed = %.2f"  
,369/1200);

---


```

# Chapter 7

## Liquid filtration

Scilab code Exa 7.1 volume of filtrate collected per cycle

```
1 clear;
2 clc;
3 printf("\n Example 7.1");
4 //In the leaf filter filtration is at const pressure
   from the start
5 //V^2 + 2ALV/v = 2(-deltaP)A^2t/(ruv)
6
7 //In the filter press ,a volume V1 of filtrate is
   obtained under const rate conditions in time t1 ,
   and filtration is then carried out at constant
   pressure .
8 //V1^2 + 2ALV1/v = 2(-deltaP)A^2t1/(ruv)
9 //and (V^2 - V1^2 ) + 2AL/ (V - V1) = 2( - P )A
   ^2/ r   (t - t1)
10
11 //for the leaf filter
12 t2 = 300;           //t2 is in secs
13 V2 = 2.5*10^(-4);  //V2 is in m^3
14 t3 = 600;           //t3 is in secs
15 V3 = 4*10^(-4);    //V3 is in m^3
16 A = 0.05;          //A is in m^2
```

```

17 deltaP = -7.13*10^(4); // it is in N/m^2
18 // putting these values in above eq
19
20 a = [2*7.13*10^(4)*0.05^(2)*300 -2*0.05*2.5*10^(-4)
       ;2*7.13*10^(4)*0.05^(2)*600 -2*0.05*4*10^(-4)];
21 b = [(2.5*10^(-4))^2;(4*10^(-4))^2];
22 x = inv(a)*b;
23 y = [1/x(1);x(2)];
24 printf("\n L/ =%f*10^(-3) and r = %f*10^(11)"
        ,y(2)*10^3,y(1)*10^(-11));
25
26 //for the filter press
27 V1 = poly([0], 'V1');
28 s = roots(V1^2 + (2.16*y(2)*V1)-(4*10^(5)*2.16^2)/y
            (1)*180);
29 printf("\n the value of V1 = %fm^3",s(2));
30
31 //For a constant pressure period (t - t1)=900 secs
32 //Calculting the total volume of filtrate
33 V = poly([0], 'V');
34 d = roots((V^2-3.33*10^(-4))+(1.512*10^(-2)*(V
            -1.825*10^(-2))-5.235*10^(-6)*900));
35 printf("\n The value of V = %.3f m^3",d(2));
36
37 f = (4*10^(5)*(2.16)^2)/(7.13*10^(11)*(6.15*10^(-2)
            + 2.16*3.5*10^(-3)));
38 printf("\n The final rate of filtration is %.2f
            *10^(-5) m^3/sec",f*10^(5));
39
40 // Assuming viscosity of the filtrate is the same
        as that of the wash-water
41 rw_400 = (0.25)*f;
42 printf("\n Rate of washing at 400 kN/m2 = %.1f
            *10^(-6) m^3/sec",rw_400*10^(6));
43
44 rw_275 = rw_400*(275/400);
45 printf("\n Rate of washing at 275 kN/m^2 = %.1f
            *10^(-6) m^3/sec",rw_275*10^6);

```

```
46 printf("\n Thus the amount of wash-water passing in  
600s = %.3f m^3", 600*rw_275);
```

---

### Scilab code Exa 7.2 Effect on optimum thickness of the cake

```
1 clear;  
2 clc;  
3 printf("\n Example 7.2");  
4 //The slurry contains 100kg whiting/m^3 of water  
5 printf("\n Volume of 100 kg whiting = %f m^3"  
     ,100/3000);  
6 printf("\n Volume of cake = %f m^3", 0.0333/0.6);  
7 printf("\n Volume of liquid in cake = %f m^3"  
     ,0.05556*0.4);  
8 printf("\n Volume of filtrate = %.3f m^3", (1-0.0222)  
     );  
9 printf("\n volume of cake/volume of filtrate v = %f"  
     ,0.0556/0.978);  
10 A = 10^(-4); //area in sq meters  
11 deltaP = -1.65*10^(5); //P is in pascals  
12 l = 0.01; //length is in meters  
13 vol_flow_rate = 2*10^(-8); //Volume flow  
    rate is in m^3/sec  
14 u = 10^(-3); //vicosity is in Ns/m^2  
15  
16 r = poly([0], 'r');  
17 r1 = roots((10^4)*(2*10^(-8)*r) - 1.65*10^(5)/(10^(-5)  
    ));  
18 printf("\n r = %.2f * 10^(13) / m^2", r1*10^(-13));  
19  
20 function [Lopt]=optimum()  
21     Lopt = 1.161*10^(-3)*(900)^(0.5); //t = 900  
        secs  
22     funcprot(0);  
23 endfunction
```

```

24 printf("\n optimum frame thickness = %.1f mm" ,2*
          optimum()*1000);
25
26 //total cycle time = 1.015L^2 + 900
27 //rate of cake production R = L/(1.015L^2 + 900)
28
29 L = poly([0] , 'L');
30 L1 = roots(1.025*10^(6)*L^2 + 900 - 2.050*10^(6)*L
            ^2);
31 printf("\n Frame thickness = %.2f mm" ,2*L1(1)*10^3)
      ;

```

---

**Scilab code Exa 7.3** Time taken to produce 1 m<sup>3</sup> of filtrate and pressure in this time

```

1 clear;
2 clc;
3 printf("\n Example 7.3");
4 V = 0.094;           //volume in m^3
5 deltaP = -3530;     //P is in kN/m^2
6
7 //At t = 1105 secs
8 V1 = 0.166;          //V is in m^3
9 deltaP1 = -5890;     //P is in kN/m^2
10
11 a = [2.21*10^(6) -0.094;6.51*10^(6) -0.166];
12 b = [0.0088;0.0276];
13 x = inv(a)*b;
14 y = [x(2);x(1)];
15 printf("\n LA/v =%f           A^2/ r v = %f*10^(-7)" ,y
        (1),y(2)*10^7);
16 printf("\n For the full size plant:");
17 printf("\n LA/v = %f           A^2/ r v=%f*10^(-7)" ,10*y
        (1),y(2)*10^8);
18

```

```

19 // Solving LHS of the integral
20 LHS = integrate('b+0.154+2.31', 'b', 0, 1);
21 //Equating LHS = RHS
22 t = LHS/(3.46*10^(-3));
23 printf("\n t = %d secs", t);
24 printf("\n deltaP = %dkN/m^2", (1+0.154)
    /(4.64*10^(-7)*857));

```

---

**Scilab code Exa 7.4** Speed of rotation for maximum throughput

```

1 clear;
2 clc;
3 printf("\n Example 7.4");
4 a = [2*84300*0.02^(2)*60
    -2*0.02*0.0003; 2*84300*0.02^(2)*120
    -2*0.02*0.00044];
5 b = [0.0003^2; 0.00044^2];
6 x = inv(a)*b;
7 y = [x(2); 1/x(1)];
8 printf("\n L/v = %f*10^(10)", y(1), y
    (2)*10^(-10));
9 printf("\n Area of filtering surface = %f m^2", 4*(
    %pi));
10 printf("\n Bulk volume of cake deposited =%.3f m^3/
    revolution", 4*(%pi)*0.005);
11
12 V = sqrt(1*10^(-6)*143^2);
13 printf("\n V = %.3f m^3", V);
14
15 t = poly([0], 't');
16 t1 = roots(0.141^2 + 2*2.19*10^(-3)
    *0.141-2*84300*(4*(%pi))^(2)*t/(3.48*10^10));
17 printf("\n t = %f secs", t1);
18 printf("\n time for 1 revolution =%.1f secs", t1/0.4)
    ;

```

```

19 printf("\n speed = %.3fHz",0.4/t1);
20 printf("\n rate of filtrate production w = %.2f kg/
sec",143/67.3)
21 printf("\n mass of slurry S =%.1f kg/sec",1.66*2.11)
;
```

---

### Scilab code Exa 7.5 Optimum filtration time for maximum throughput

```

1 clear;
2 clc;
3 printf("\n Example 7.5");
4 V1 = 0.00025; //V is in m^3
5 t = 300; //t is in secs
6 a = [7.14*10^(-6) 2.86*10^(-4);11.42*10^(-6)
2.86*10^(-4)];
7 b = [1.2*10^(6);1*10^(6)];
8 x = inv(a)*b;
9
10 //for the plate and frame filter
11 B1 = x(1)/(2*2.2^2*413*10^3);
12 B2 = x(2)/(2.2*413*1000);
13
14 printf("\n r v = %d\n",x(1));
15 printf("\n r l = %d",x(2));
16 printf("\n B1= %f B2= %f",B1,B2);
17 printf("\n the filtration time for maximum
throughput is:");
18 t1 = 21.6*10^3;
19 t0= t1 +B2*(t1/B1)^(0.5);
20 printf("\n t = %f secs",t0);
21 V = (t1/B1)^(0.5);
22 printf("\n V= %f m^3",V);
23 printf("\n Mean rate of filtration is: %.2f *10^-6 m
^3/s", (V/(t1+t0))/10^-6);
```

---

### Scilab code Exa 7.6 Thickness of cake produced

```
1 clear;
2 clc;
3 printf("\n Example 7.6");
4 A=0.6*0.6*%pi;           //in m^2
5 rate=1.25*10^-4;         // in m^3/s
6
7 v_w=0.2/(3*10^3);
8 v_f=10^-3-v_w;
9
10 v=v_w/v_f;
11 v_rate=rate*v;
12 w=360*0.2;
13
14 t=v_rate*w/A;
15 printf("\nThickness of cake produced is : %.1f mm",t
    /10^-4);
16 K = poly([0], 'K');
17 K1 = roots((1.25*10^(-4)*360)^2-K*(6.5*10^(4)
    *(0.36*(%pi))^(2)*72));
18 printf("\n The value of K is %.2f*10^(-10)",K1
    *10^(10));
19
20 //Filter press
21 //Using a filter press with n frames of thickness b
    m the total time , for one complete cycle of the
    press =(tf+120n+240), where tf is the time during
    which filtration is occurring
22 //overall rate of filtration = Vf/( tf + 120n + 240)
23
24 // Vf = 0.3^(2)*n*b/0.143
25 // tf = 2.064*10^5 b^2
26
```

```

27 b = poly([0], 'b');
28 b1 = roots(b^2 - 0.0458*b - 0.001162);
29 printf("\n The thickness is %.4f m", b1(1));
30
31 function[n]=number_of_plates()
32
33     n = (0.030 + 25.8*b1(1)^2)/(0.629*b1(1)-0.015);
34     funcprot(0);
35 endfunction
36 n = number_of_plates();
37 printf("\n The minimum number of plates required is
38 %d", ceil(n));
39 d = poly([0], 'd');
40 d1 = roots(ceil(n)*(0.629*d-0.015)-0.030-25.8*d^2);
41 printf("\n The sizes of frames which will give
42 exactly the required rate of filtration when six
43 are used are %f mm", d1*10^3);
44 printf("\n\n Thus any frame thickness between 47
45 and 99 mm will be satisfactory. In practice ,50 mm
46 (2 in) frames would probably be used .")

```

---

**Scilab code Exa 7.7** Increase in the overall throughput of the press

```

1 clear;
2 clc;
3 printf("\n Example 7.7");
4 //Case 1
5
6 //dV/dt = A^2(-deltaP)/vru(V + AL/v)
7 //For constant rate filtration:
8 //Vo/to = a/Vo + b
9 //Vo^2 + bVo = ato
10 //For constant pressure filtration

```

```

11 // 0.5(V^2 - Vo^2)+b(V-Vo)=a(t-to)
12 // to=600s , t-to=3600s , Vo=V/4
13 // V^2/16 +bV/4 = 600a
14 // 0.5(V^2 - V^2/16)+b(V-V/4)=3600a
15 // 3600a = (15/32)V^2 +3/4(bV) =
            3/8(V^2) + 3/2(bV)
16 // b = V/8
17 // a = (V^2/16 + V^2/32)/600 =
            (3/19200)V^2
18 // Total cycle time = 900 + 4200 =
            5100 secs
19 // Filtration rate = V/5100 =
            0.000196V
20
21 // Case 2
22 // V1/t1 = a / (V1 +b/4)=Vo/to=a/(Vo+b
23 // )  

24 // 0.5*(49/64V^2 - V1^2)+b/4(7/8V-V1)
25 // =a(t-t1)
26 // V/2400 = (3/19200)V^2/(V1+V/32)
27 // t1 = (to/Vo)V1
28 t1 = 600/(1/4)*(11/32);
29 printf("\n t1 = %dsecs",t1);
30 // Substituting gives
31 deltaT = (19200/3)*(784-121+34)/2048;
32 printf("\n t -t1 = %d secs",deltaT);
33 Cycle_time = 180+900+t1+deltaT;
34 printf("\n cycle time = %d secs",Cycle_time);
35 Increase = (0.000214 - 0.000196)/(0.000196)*100;
36 printf("\n Increase in filtration rate is %.1f per
            cent",Increase);

```

---

# Chapter 8

## Membrane Separation Processes

Scilab code Exa 8.2 Area of membrane and average flux

```
1 clear;
2 clc;
3 printf("\n Example 8.2");
4
5 //From the gel polarisation model:
6 //J = (1/A)(dV/dt) = hD ln(Cg/Cf)
7 //Cf = Co(Vo/V)
8 // where Co and Vo are the initial concentration
// and volume, respectively and Cf and V are the
// values at subsequent times
9 //Combining these eq gives
10 //dV/dt = A(hDln(Cg/Co)-hDln(Vo/V))
11 V = [10 5 3 2 1];
12 y = [9.90 13.64 18.92 27.30 112.40];
13 plot(V,y)
14 xtitle("Area under the curve is 184.4","Volume(m^3)"
,"( J - hDln(Vo/V) ) ^(-1)" )
15
16
```

```

17 // (b)
18 Jo = 0.04*log(250/20);
19 printf("\n Jo = %.3f m/h", Jo);
20 Jf = 0.04*log(250/200);
21 printf("\n Jf = %f m/h", Jf);
22 Jav = Jf + 0.27*(0.101-0.008);
23 printf("\n Jav = %f m/h", Jav);
24 //For the removal of 9m^3 filtrate in 4 hours
25 Area = (9/4)/Jav;
26 printf("\n Area = %fm^2", Area);

```

---

**Scilab code Exa 8.3** Minimum number of membrane modules required

```

1 clear;
2 clc;
3 printf("\n Example 8.3");
4 //It is assumed that Q0 is the volumetric flowrate
   of feed
5 // Q2 the volumetric flowrate of concentrate
6 //C0 the solute concentration in the feed
7 // C2 the solute concentration in the concentrate
8 // F the volumetric flowrate of membrane permeate
9 // A the required membrane area.
10 // It is also assumed that there is no loss of
    solute through the membrane.
11 C1 = 3;
12 while 1
13     C1new = C1 -(0.04-0.02*log(30/C1))/(C1^(-1)/50);
14     if C1new == C1 then
15         break;
16     end
17     C1 = C1new;
18 end
19 printf("\n C1 = %d kg/m^3", C1);
20 printf("\n below this concentration the membrane")

```

```

        flux is 0.04 m/h");
21
22 //This does not pose a constraint for the single
   stage as the concentration of solute C2 will be
   that of the final concentrate , 20 kg/m3.
23 //Conservation of solute gives:QoCo = Q2C2
24 //A fluid balance gives : Qo = F + Q2
25 //Combining these eq and substituting Known values:
26 A = (2.438/0.02)/log(30/20);
27 printf("\n A = %d m^2",A);
28 //The tubular membranes to be used are available as
   30 m^2modules.
29 printf("\n the no of required modules are %d ",A/30)
   ;
30
31 //Part(b)
32
33 //Conservation of solute gives = QoCo = Q1C1 = Q2C2
34 //A fluid balance on stage 2 gives Q1 = Q2 + F2
35 //A fluid balance on stage 2 gives Q1 = Q2 +F2
36 //Substituting given values in above eqns
37 //2.5 = 1.25/C1 + 0.02A1ln(30/C1)
38 function[A1]=a(C1)
39     A1 = (2.5-1.25/C1)/(0.02*log(30/C1));
40     funcprot(0);
41 endfunction
42 function[A2]=b(C1)
43     A2 = (1.25/C1 - 0.0625)/0.00811
44     funcprot(0);
45 endfunction
46
47 printf("\n The procedure is to use trial and error
   to estimate the value of C1 that gives the
   optimum values of A1 and A2");
48 printf("\n If C1 = 5kg/m^3 then A1 = %d m^2 and A2 =
   %d m^2",a(5),b(5));
49 printf("\n an arrangement of 3 modules      1 module
   is required.");

```

```
50 printf("\n\n\n If C1 = 4 kg/m^3 then A1 = %dm^2 and  
      A2 = %dm^2",a(4),b(4));  
51 printf("\n an arrangement of 2 modules 1 module  
      is almost sufficient.");  
52 printf("\n\n\n If C1 = 4.5 kg/m^3 then A1 = %dm^2  
      and A2 = %d m^2",a(4.5),b(4.5));  
53 printf("\n an arrangement of 2 modules 1 module  
      which meets the requirement");  
54 printf("\n\n This arrangement requires the minimum  
      number of modules.");
```

---

# Chapter 9

## Centrifugal Separations

Scilab code Exa 9.1 Value of capacity factor

```
1 clear;
2 clc;
3 printf("\n Example 9.1");
4 d_particle = 5;           // particle size is in um
5 p = 1000;                 // density of water in kg/m^3
6 ps = 2800;                // density of solids in kg/m
                            ^3
7 viscosity = 10^(-3);      // viscosity is in Ns/m^2
8 uo = ((d_particle*10^(-6))^2)*(ps-p)*9.81/(18*
viscosity);
9 printf("\n Terminal falling velocity of particles of
diameter = %.2f m/sec",uo*10^5);
10 Q = 0.25;                // volumetric flow rate is in
                            m^3/sec
11 printf("\n E = %.2f *10^(4) m^2", (Q/uo)*10^(-4));
12
13 printf("\n For coal-in-oil mixture");
14 uo1 = 0.04/(Q/uo);
15 printf("\n uo = %.2f *10^-6 m/sec", uo1*10^6);
16
17 d = sqrt((18*0.01*uo1)/((1300-850)*9.81));
```

```
18 printf("\n d = %d um", (d/3)*10^6);
```

---

### Scilab code Exa 9.3 Time taken to produce filtrate

```
1 clear;
2 clc;
3 printf("\n Example 9.3");
4 //In the filter press
5 // V^2 + 2(AL/v)V = 2(-deltaP)A^2*t/(ruv)
6
7 l = 0.025;           //l is in meters
8 L = 0.003;           //L is in meters
9 deltaP = 350;        //it is in N/m^2
10 t = 3600;           //t is in secs
11
12 // x = v/ru
13 x = poly([0], 'x');
14 x1 = roots(0.025^2 + 2*0.003*0.025 - 2*3.5*10^(5)
   *3600*x);
15 printf("\n the value of ru/v = %.2f*10^12", (1/x1)
   *10^(-12));
16
17 //In the centrifuge
18 R = 0.15;           //R is in meters
19 H = 0.20;           //H is in meters
20 V = 0.00225;        //V is in m^3
21 r = poly([0], 'r');
22 r1 = roots(%pi*(R^2 - r^2)*H-V);
23 printf("\n Value of ro = %f mm", r1(1)/2);
24 printf("\n angular frequency = %.1 frad/s", (r1(1)
   /2*10^3)*2*(%pi));
25
26 // (R^2 -r^2)(1+2L/R)+2r^2 ln ( r/R) = 2vtpw^2/ru (R^2-ro
   ^2)
27 t = poly([0], 't');
```

```
28 t1 = roots((R^2 - r1(1)^2)*(1+2*(L/R))+2*(r1(1)^2)*  
    log(r1(1)/R)-2*t*1000*408.4^(2)/(3.25*10^12)*(R  
    ^2-(r1(1)/2)^2));  
29 printf("\n time required = %f secs",t1);
```

---

# Chapter 10

## Leaching

**Scilab code Exa 10.1** Time required for solute to dissolve

```
1 clear
2 clc;
3 printf("\n Example 10.1");
4 //For the pilot scale vessel
5 c = (2.5*75)/100;           //in kg/m^3
6 cs = 2.5;                   //in kg/m^3
7 V = 1.0;                     //V is in m^3
8 t = 10;                      //t is in secs
9
10 // $1.875 = 2.5(1 - e^{(-kA/b * 100)})$ 
11 // x = kA/b
12 x = -log(1 - 1.875/2.5)/10;
13
14 //For the full scale vessel
15 c = (500*28/100)/100;
16 printf("\n C = %fkg/m^3", c);
17 cs = 2.5;                   //cs is in kg/m^3
18 V = 100;                     //V is in m^3
19 t = -log(1 - 1.4/2.5)*(100/0.139); //t is in secs
20 printf("\n t = %d secs", t);
```

---

### Scilab code Exa 10.2 Rate of feed of neutral water to the thickeners

```
1 clear;
2 clc;
3 printf("\n Example 10.2");
4
5 // If x1, x2, x3 are the solute: solvent ratios in
   thickeners 1, 2, and 3, respectively, the
   quantities of CaCO3, NaOH, and water in each of
   the streams can be calculated for every 100 kg of
   calcium carbonate
6
7 // Since the final underflow must contain only 1 per
   cent of NaOH
8 function [f]=F(x)
9   f(1)=(300*x(3))/100 - 0.01;
10  f(2)=300*(x(2)-x(3))/x(4) - x(3);           //Wf =
11  x(4)
12  f(3)=300*(x(1)-x(3))/x(4) - x(2);
13  f(4)=(80-300*x(3))/(600+x(4))-x(1);
14  funcprot(0);
15 endfunction
16 //An initial guess
17 x = [0.1 0.1 0.1 0.1];
18 y = fsolve(x,F);
19 printf("\n x1 = %f x2 = %f x3 = %f Wf = %f" ,y(1),y
20 (2),y(3),y(4));
21 printf("\n Thus the amount of water required for
   washing 100 kg CaCO3 is %f kg" ,y(4));
22 printf("\n The solution fed to reactor contains 0.25
   kg/s Na2CO3. This is equivalent to 0.236 kg/s
   CaCO3, and hence the actual water required is %.2 f
   kg/sec" ,y(4)*0.236/100);
```

---

### Scilab code Exa 10.3 Required number of thickeners

```
1 clear;
2 clc;
3 printf("\n Example 10.3");
4
5 //part 1
6 //Solvent in underflow from final washing thickener
= 50 kg/s.
7 //The solvent in the overflow will be the same as
that supplied for washing (200 kg/s).
8 //Solvent discharged in overflow/Solvent discharged
in underflow= for the washing thickeners.
9 //Liquid product from plant contains 54.9 kg of salt
in 195 kg of solvent.
10 //This ratio will be the same in the underflow from
the first thickener.
11 printf("\n the material fed to the washing
thickeners consists of 100 kg TiO2, 50 kg solvent
and %d kg salt",50*(54.9/195));
12
13 //m = n+1
14 m = log(421)/log(4);
15 printf("\n The required number of thickeners for
washing are %f",m);
16
17 //Part 2
18 //From an inspection of the data, it is seen that Wh
= 0.30 + 0.2Xh.
19 //Thus: Sh = WhXh = 0.30Xh + 0.2X2h= 5Wh2 - 1.5Wh
20 //Considering the passage of unit quantity of TiO2
through the plant, then:
21 Ln = 0; wn = 2; Xn = 0;
22 //since 200 kg/s of pure solvent is used.
```

```

23 Sn = 0.001;
24 Wn = 0.3007;
25 So = 0.55;
26 Wo = 1.00;
27 //X1 = (Ln+1 + S0      Sn)/(wn+1 + W0      Wn)
28 X1 = (0+0.55-0.001)/(2+1-0.3007);
29 printf("\n concentration in the first thickener is
          %f",X1);
30
31 W1 = 0.30+0.2*0.203;
32 printf("\n W1 = %f",W1);
33 S1 = (0.3406*0.203);
34 printf("\n S1 = %f",S1);
35 X2 = (0.0691 - 0.001)/(1.7 + 0.3406);
36 printf("\n X2 = %f",X2);
37 W2 = (0.30+0.2*0.0334);
38 printf("\n W2 = %f",W2);
39 X3 = (0.01025-0.001)/(1.7+0.3067);
40 printf("\n X3 = %f",X3);
41 printf("\n W3 = 0.30089");
42 printf("\n S3 = %f",0.0013);
43 printf("\n W4 = %f and S4 = %f",0.30003,0.000045);
44 printf("\n Thus S4 is less than Sn and therefore 4
          thickeners are required.");

```

---

#### Scilab code Exa 10.4 Number of ideal stages required

```

1 clear;
2 clc;
3 printf("\n Example 10.4");
4 //Since the seeds contain 20 per cent of oil
5 xAo=0.2;
6 xBo=0.8;
7 printf("\n xAo = %.1f and xBo = %.1f",xAo,xBo);
8 //The final solution contains 50 per cent oil

```

```

9  yA1=0.5;
10 yS1=0.5;
11 printf("\n yA1 = %.1f and yS1 = %.1f",yA1,yS1);
12 //The solvent which is used for extraction is pure
   and hence
13 ySn1=1;
14 //1 kg of insoluble solid in the washed product is
   associated with 0.5 kg of solution and 0.025 kg
   oil.
15 xAn = 0.0167;
16 xBn = 0.6667;
17 xSn = 0.3166;
18
19 //The mass fraction of insoluble material in the
   underflow is constant and equal to 0.667. The
   composition of the underflow is therefore
   represented, on the diagram Figure 10.22, by a
   straight line parallel to the hypotenuse of the
   triangle with an intercept of 0.333 on the two
   main axes.
20
21 //The difference point is now found by drawing in
   the two lines connecting x0 and y1 and xn and yn
   +1.
22
23 //The graphical construction described in the text
   is then used and it is seen from Figure 10.22
   that xn lies in between x4 and x5.
24 printf("\n Thus 5 thickeners are adequate and for
   the required degree of extraction");

```

---

### Scilab code Exa 10.5 Number of theoretical stages required

```

1 clear;
2 clc;

```

```

3 printf("\n Example 10.5");
4
5 //On the basis of 100 kg untreated solids
6 //In the underflow feed:
7 //0.35 kg oil is associated with each kg of
     exhausted livers.
8 printf("\n mass of livers fed=%d kg containing %d kg
           oil",100/(1+0.35),100-74);
9 xA = 0.26;
10 printf("\n xA = %.2f",xA);
11 xs = 0;
12 //In the overflow feed, pure ether is used
13 ys = 1.0;
14 xs = 0;
15 //Recovery = 90 per cent
16 printf("\n          Exhausted livers
             Oil           Ether");
17 printf("\n Underflow feed           %d
             %d           -",74,26);
18 printf("\n Overflow feed           -
             -           %d",50);
19 printf("\n Underflow product       %d
             %d           e (say)",74,2.6);
20 printf("\n Overflow product       -
             -           50-e",23.4);
21 printf("\n In the underflow product:");
22 printf("\n the ratio (oil/exhausted livers) = %.3f kg
           /kg",2.6/74);
23 printf("\n Ratio (ether/exhausted livers) = %.3f kg/
           kg",0.306);
24 printf("\n e = %.1f kg",0.306*74);
25 //In the overflow product:
26 printf("\n The mass of ether = %.1f kg",50-22.6);
27 printf("\n yA = %.2f",23.4/(23.4+27.4));
28 printf("\n ys = %.2f",1-23.4/(23.4+27.4));
29 printf("\n ");

```

---

# Chapter 11

## Distillation

**Scilab code Exa 11.1** Mole fraction calculation

```
1 //Example 11.1
2
3 clear;
4 clc;
5
6 printf("\tExample 11.1\n");
7
8 p0_A=106; //Vapour pressure of n-heptane in kN/m^2
9 p0_B=73.7; //Vapour pressure of toluene in kN/m^2
10 P=101.3; //Total pressure in kN/m^2
11
12 xA=(P-p0_B)/(p0_A-p0_B);
13
14 yA=p0_A*xA/P;
15
16 printf("\nMole fraction in liquid phase is : %.3f",
17     xA);
17 printf("\nMole fraction in vapour phase is : %.3f\n"
18     ,yA);
19 //End
```

---

### Scilab code Exa 11.2 Saturated Pressure calculation

```
1 //Example 11.2
2
3 clear;
4 clc;
5
6 printf("\tExample 11.2\n");
7
8 Pc=4700; //Critical pressure in kN/m^2
9 Tc=508.1; //critical temperature in K
10 p1=100.666; //in kN/m^2
11 T1=329.026; //in K
12 T=350.874; //in K
13
14 Tr1=T1/Tc;
15 Pr1=p1/Pc;
16 printf("\nTr1 = %f \nPr1 = %f\n",Tr1,Pr1);
17
18 c5=-35+(36/Tr1)+(42*log(Tr1))-(Tr16);
19 c2=((0.315*c5)-log(Pr1))/((0.0838*c5)-log(Tr1));
20 c1=0.0838*(3.758-c2);
21 printf("\nc5 = %.4 f \nc2 = %.4 f \nc1 = %.4 f\n",c5,c2
, c1);
22
23 k9=-35*c1;
24 k10=-36*c1;
25 k11=(42*c1)+c2;
26 k12=-c1;
27
28 printf("\nk9 = %.3 f \nk10 = %.3 f \nk11 = %.4 f",k9,
k10,k11);
29 printf("\nk12 = %.5 f\n",k12);
30
```

```

31 Tr=T/Tc;
32 Pr=exp(k9-(k10/Tr)+(k11*log(Tr))+(k12*Tr^6));
33 p0=Pc*Pr;
34 printf("\nPr = %f \nP0 = %.2 f kN/m^2\n",Pr,p0);
35
36 //End

```

---

**Scilab code Exa 11.3** Vapour phase composition of a mixture

```

1 //Exaple 11.3
2
3 clear;
4 clc;
5
6 printf("\tExample 11.3\n");
7
8 k1_B=6.90565;
9 k2_B=1211.033;
10 k3_B=220.79;
11
12 k1_T=6.95334;
13 k2_T=1343.943;
14 k3_T=219.377;
15
16 t=338-273; //in Degree celsius
17 P=101.3; //in kN/m^2
18 xB=0.5;
19 xT=0.5;
20
21 function [p0]=antoine(k1,k2,k3,T)
22     p0=10^(k1-(k2/(T+k3)));
23     funcprot(0)
24 endfunction
25
26 p0_B=antoine(k1_B,k2_B,k3_B,t)*101.325/760;

```

```

27 p0_T=antoine(k1_T,k2_T,k3_T,t)*101.325/760;
28
29 printf("\nP0_B = %.1f kN/m^2\nP0_T = %.1f kN/m^2\n",
30      p0_B,p0_T);
31 pB=xB*p0_B;
32 pT=xT*p0_T;
33 printf("\nPB = %.2f kN/m^2 \nPT = %.3f\n",pB,pT);
34
35 p=pB+pT;
36 yB=pB/p;
37 yT=pT/p;
38 printf("\nyB = %.3f \nyT = %.3f\n",yB,yT);
39
40 //End

```

---

**Scilab code Exa 11.4** Boiling point of equimolar mixture

```

1 //Example 11.4
2
3 clear;
4 clc;
5
6 printf("\tExample 11.4\n");
7
8 function[p0]=antoine(k1,k2,k3,T)
9     p0=10^(k1-(k2/(T+k3-273)));
10    funcprot(0)
11 endfunction
12
13 k1_B=6.90565;
14 k2_B=1211.033;
15 k3_B=220.79;
16
17 k1_T=6.95334;

```

```

18 k2_T=1343.943;
19 k3_T=219.377;
20
21 xB=0.5;
22 xT=0.5;
23
24 printf("\n\tT(K)");
25 T=[373 353 363 365 365.1];
26 disp(T);
27
28 i=1;
29
30 while i<6
31     p0_B(i)=antoine(k1_B,k2_B,k3_B,T(i))
32         *101.325/760;
33     p0_T(i)=antoine(k1_T,k2_T,k3_T,T(i))
34         *101.325/760;
35     pB(i)=xB*p0_B(i);
36     pT(i)=xT*p0_T(i);
37     p(i)=pB(i)+pT(i);
38     i=i+1;
39 end
40 printf("\n\tP0_B");
41 disp(p0_B);
42 printf("\n\tP0_T");
43 disp(p0_T);
44 printf("\n\tPB");
45 disp(pB);
46 printf("\n\tPT");
47 disp(pT);
48 printf("\n\tPB+PT");
49 disp(p);
50 //since total pressure at 365.1 K is nearly same as
51 // 101.3 kPa
52 printf("\nBoiling temperature is %f K",T(5));
53
54 //End

```

---

### Scilab code Exa 11.5 Dew point of a equimolar mixture

```
1 //Example 11.5
2
3 clear;
4 clc;
5
6 printf("\tExample 11.5\n");
7
8 function [p0]=antoine(k1,k2,k3,T)
9     p0=10^(k1-(k2/(T+k3-273)));
10    funcprot(0)
11 endfunction
12
13 k1_B=6.90565;
14 k2_B=1211.033;
15 k3_B=220.79;
16
17 k1_T=6.95334;
18 k2_T=1343.943;
19 k3_T=219.377;
20
21 //Since total pressure is 101.3 kPa, pB=pT
22 pB=50.65;
23 pT=50.65;
24
25 printf("\n\tT(K)");
26 T=[373.2 371.2 371.7 371.9 372];
27 disp(T);
28
29 i=1;
30
31 while i<6
32     p0_B(i)=antoine(k1_B,k2_B,k3_B,T(i))
```

```

            *101.325/760;
33      p0_T(i)=antoine(k1_T,k2_T,k3_T,T(i))
            *101.325/760;
34      xB(i)=pB/p0_B(i);
35      xT(i)=pT/p0_T(i);
36      x(i)=xB(i)+xT(i);
37      i=i+1;
38  end
39
40  printf("\n\tp0  B")
41  disp(p0_B);
42  printf("\n\tp0  T");
43  disp(p0_T);
44  printf("\n\txB");
45  disp(xB);
46  printf("\n\txT");
47  disp(xT);
48  printf("\n\tx = xB+xT");
49  disp(x);
50
51 //Since last value is closer to 1, 372 K is the
   required dew point
52 printf("\nDew point can be taken as %f K",T(5));
53
54 //End

```

---

### Scilab code Exa 11.6 Composition of vapour and liquid

```

1 // Example 11.6
2
3 clear;
4 clc;
5
6 printf("\tExample 11.6\n");
7

```

```

8 // Fractional vapourisation f=V/F
9 f=0.25;
10
11 slope=(1-f)/-f;
12
13 //from fig 11.9 , a construction is made
14 printf("\nFor slope = %d",slope);
15 printf("\nx=0.42 \t y=0.63\n");
16
17 //from fig. 11.10 , temperature corresponding to x
18 // =0.42 is noted
19 printf("\nfor x=0.42 \t T=366.5");
20 //End

```

---

### Scilab code Exa 11.7 Number of theoretical plates needed

```

1 clear;
2 clc;
3 printf('Example 11.7'); //Example 11.7
4 // Find Number of theoretical plates needed and the
5 // position of entry for the feed
6 F = 100; //Feed [kmol]
7
8 function [f]=Feed(x)
9     f(1)=x(1)+x(2)-100; // Overall
10    mass Balance
11    f(2)=0.9*x(1)+.1*x(2)-(100*.4); //A
12    balance on MVC, benzene
13    funcprot(0)
14 endfunction
15 x = [50 50];
16 product = fsolve(x,Feed);
17

```

```

16 //Using notation of figure 11.13
17 Ln = 3*product(1);
18 Vn = Ln + product(1);
19
20 //Reflux to the plate
21 Lm = Ln + F;
22Vm = Lm - product(2);
23
24 //Equilibrium Composition
25 xt = .79;      yt = .9;
26 //From Top eqm line
27 yt1 = (Ln/Vn)*xt + (product(1)/Vn);
28 xt1=.644;        //Thus from Eqm curve for yt1
29 //From Top eqm line
30 yt2 = (Ln/Vn)*xt1 + (product(1)/Vn);
31 xt2=.492;        //Thus from Eqm curve for yt2
32 //From Top eqm line
33 yt3 = (Ln/Vn)*xt2 + (product(1)/Vn);
34 xt3=.382;        //Thus from Eqm curve for yt3
35 //From II Eqm Line
36 yt4 = (Lm/Vm)*xt3 - (product(2)/Vm)*.1;
37 xt4=.2982;       //Thus from Eqm curve for yt4
38 //From II Eqm Line
39 yt5 = (Lm/Vm)*xt4 - (product(2)/Vm)*.1;
40 xt5=.208;         //Thus from Eqm curve for yt5
41 //From II Eqm Line
42 yt6 = (Lm/Vm)*xt5 - (product(2)/Vm)*.1;
43 xt6=.120;         //Thus from Eqm curve for yt6
44 //From II Eqm Line
45 yt7 = (Lm/Vm)*xt6 - (product(2)/Vm)*.1;
46 xt7=.048;         //Thus from Eqm curve for yt7
47
48 //Equilibrium Data
49 y=[0 yt7 yt6 yt5 yt4 yt3 yt2 yt1 yt];
50 x=[0 xt7 xt6 xt5 xt4 xt3 xt2 xt1 xt];
51 //Top Equilibrium Line equation 11.35
52 x1 = linspace(0,.79,100);
53 y1 = (Ln/Vn)*x1 + (product(1)/Vn);

```

```

54 // Equilibrium Line equation 11.37
55 x2 = linspace(0.048,.44,100);
56 y2 = (Lm/Vm)*x2 - (product(2)/Vm)*.1;
57 clf();
58 plot(x,y,x1,y1,x2,y2);
59 xtitle("Lewis-Sorel Method", "Mole fraction of C6H6
      in Liquid (x)", "Mole Fraction C6H6 in Vapor (y)"
      );
60 legend ("Equilibrium Plot", "Top Eqm Line", "Bottom
      Eqm Line");
61 printf("\n\n As the least point on equilibrium Line
      xt-7 correspond to reboiler , and there will be
      seven plates");
62
63
64 //END

```

---

### Scilab code Exa 11.8 The McCabe Thiele method

```

1 clear;
2 clc;
3 printf('Example 11.8'); //Example 11.8
4 // Find Number of theoretical plates needed and the
   position of entry for the feed by McCabe Thiele
   method
5
6 F = 100; //Feed [kmol]
7
8 function [f]=Feed(x)
9     f(1)=x(1)+x(2)-100; //Overall
                           mass Balance
10    f(2)=0.9*x(1)+.1*x(2)-(100*.4); //A
                                             balance on MVC, benzene
11    funcprot(0)
12 endfunction

```

```

13 x = [50 50];
14 product = fsolve(x,Feed);
15
16 //Using notation of figure 11.13
17 Ln = 3*product(1);
18 Vn = Ln + product(1);
19
20 //Reflux to the plate
21 Lm = Ln + F;
22Vm = Lm - product(2);
23
24 //Equilibrium Data
25 y=[0 .127 .252 .379 .498 .594 .708 .818 .9 1];
26 x=[0 .048 .12 .208 .298 .382 .492 .644 .79 1];
27 //Diagnol Line
28 y3 = [0 1];
29 x3 = [0 1];
30 //Top Equilibrium Line equation 11.35
31 x1 = linspace(0,.985,100);
32 y1 = (Ln/Vn)*x1 + (product(1)/Vn);
33 //Equilibrium Line equation 11.37
34 x2 = linspace(0.048,.44,100);
35 y2 = (Lm/Vm)*x2 - (product(2)/Vm)*.1;
36 clf();
37 //Setting initial point A x = .985 at top eqm line
38 xm = [.985 .965 .965 .92 .92 .825 .825 .655 .655 .44
        .44 .255 .255 .125 .125 .048];
39 ym = [.985 .985 .965 .965 .92 .92 .825 .825 .655
        .655 .44 .44 .255 .255 .125 .125];
40 xp = [.985 .965 .92 .825 .655 .44 .255 .125 .048];
41 yp = [.985 .965 .92 .825 .655 .44 .255 .125 .048];
42 plot(x,y,x3,y3,x1,y1,x2,y2,xm,ym);
43 xtitle("McCabe Thiele Method", "Mole fraction of
    C6H6 in Liquid (x)", "Mole Fraction C6H6 in Vapor
    (y)");
44 legend ("Equilibrium Plot", "Diagnol Line", "Top Eqm
    Line", "Bottom Eqm Line",5);
45 xset('window',1);

```

```

46 for(i=2:8)
47 plot(xp(i),yp(i),"o-");
48 xtitle("Equilibrium plot","mole fraction C6H6 in
        liquid(x)","mole fraction C6H6 in vapour(y)");
49 end
50 printf("\n\n The Number of stages are then counted
        highlighted points that is number of plates
        required as 7");
51
52
53 //END

```

---

### Scilab code Exa 11.9 Number of plates required at total reflux

```

1 //Example 11.9
2
3 clear;
4 clc;
5
6 printf("\tExample 11.9\n");
7
8 xA_d=0.9;
9 xA_s=0.1;
10 xB_d=1-xA_d;
11 xB_s=1-xA_s;
12
13 a=2.4;
14 xd=0.9;
15 xf=0.4;
16 xw=0.1;
17
18 n=(log(xA_d*xB_s/(xB_d*xA_s))/log(a))-1;
19 printf("\nNo. of theoretical plates in column is :
        %d",n);
20

```

```

21 Rm=((xd/xf)-(a*((1-xd)/(1-xf)))/(a-1);
22 printf("\nMinimum reflux ratio is %.2f\n",Rm);
23
24 yf=0.61;
25 printf("\nUsing Graphical construction with yf=0.61\
n");
26 Rmin=(xd-yf)/(yf-xf);
27 printf("Minimum reflux ratio is %.2f\n",Rmin);
28
29 //End

```

---

### Scilab code Exa 11.10 Heat required

```

1 //Example 11.10
2
3 clear;
4 clc;
5
6 printf("\tExample 11.10\n");
7
8 //From material balance
9 // D+W=1
10 // 0.995D+0.1W=1*3
11
12 A=[1 1;0.995 0.1];
13 B=[1;3];
14 Rm = (1952-1547)/(1547-295);
15 printf("\n Rm = %.3f",Rm);
16 NA = 1.08*405;
17 printf("\n Since the actual reflux is 8 pre cent
above the minimum NA = 1.08*NmA = %.3f",NA);
18 N = 5/0.6;
19 printf("\n Number of plates to be required are %.3f
",5/0.6);
20

```

```

21 Qb_W = 582 - (-209);
22 printf("\n Heat input to the boiler per unit mass of
           bottom product is %.3f",Qb_W);
23 printf("\n Heat input to the boiler = %.3f kW"
           ,791*0.78);
24 printf("\n Condenser duty = %d kW", (1984-296)*0.22);

```

---

### Scilab code Exa 11.11 Number of theoretical stages required

```

1 //Example 11.11
2
3 clear;
4 clc;
5
6 printf("\tExample 11.11\n");
7
8 //      F is feed
9 //      D is distillate
10 //      W is waste
11 //      S is sidestream
12
13 F=100;
14 S=10;
15
16 //Mass fractions of CCl4 in various streams
17 xf=0.5;
18 xd=0.95;
19 xw=0.05;
20 xs=0.8;
21
22 //      D + W = 100-10
23 //      0.95D + 0.00W = 50-8
24 A=[1,1;0.95,0.05];
25 B=[90;42];
26 x=inv(A)*B;

```

```

27 printf("\nD = %.1f      W = %.1f\n",x(1),x(2));
28
29 disp("From the enthalpy data and the reflux ratio ,
       the upper pole point M is located as shown in
       Figure .");
30
31 disp("Points F and S are located ,such that FS/FF =
       10.");
32
33 disp("MF is Joined and extended to cut NS^A at O,
       the immediate pole point .");
34 disp("The number of stages required is then obtained
       from the figure and ");
35 printf("13 theoretical stages are required");
36
37 //End

```

---

### Scilab code Exa 11.12 Amount of distillate

```

1 clear;
2 clc;
3 printf("\n Example 11.12");
4 R = [0.85 1.0 1.5 2.0 3.0 4.0];      //Reflux ratio
5 xd = 0.75;                           //top
                                         concentration of alcohol
6 xs = [0.55 0.50 0.37 0.20 0.075 0.05]; //From the
                                         graph fig.11.35 page-596
7 Db(1) = 0;
8
9 printf("\n      R                  Fi          xs
           Db      " );
10 i=1;
11 while i<=6
12     Fi(i) = xd/(R(i) + 1);
13     if i>1 then

```

```

14     Db(i) = 100*(xs(1)-xs(i))/(xd-xs(i));
15     end
16     printf("\n %.2f          %.3f          %.2f      %"
17           .1f",R(i),Fi(i),xs(i),Db(i));
17     i=i+1;
18 end
19 plot(R,Db);
20 xtitle("", "Reflux ratio (R)", "Product Db (kmol)");
21 printf("\n The area under the Db vs R curve is given
22 by 96 kmol");
22 Hav = 4000;           // average latent heat in kJ/
23 kmol
23 Qr = 96*Hav/1000;
24 printf("\n Heat to be supplied to provide the reflux
25 ,Qr is approximately %.1f MJ",Qr);
25 printf("\n Heat to be supplied to provide the reflux
26 per kmol of product is then %.2f MJ",380/71.4);
26 printf("\n Total heat =%.2f MJ/kmol product"
27 ,5.32+4.0);

```

---

### Scilab code Exa 11.13 Heat required and average composition

```

1 //Example 11.13
2
3 clear;
4 clc;
5
6 printf("\tExample 11.13\n");
7
8 xs=[0.55;0.5;0.425;0.31;0.225;0.105];
9 xd=[0.78;0.775;0.77;0.76;0.75;0.74];
10 differ=xd-xs;
11 for i=1:6
12     reci(i)=1/(xd(i)-xs(i));
13 end

```

```

14
15 m=[xs xd differ reci];
16 printf("\n      xs      xd      (xd-xs)    1/(xd-xs)\n"
17   );
17 disp(m);
18 plot(xs,reci);
19 xtitle('Graphical integration ', 'xs ', '1/xd-xs ')';
20
21 //Area under the curve is calculated and is found to
22 //be 1.1
22
23 //logdiv = S1/S2 = area under the curve
24 logdiv=1.1;
25 S1=100;           //Assume
26 div=exp(logdiv);
27 S2=S1/div;
28 Db=S1-S2;        //Product obtained
29 amt=xs(1)*S1-(xs(6)*S2);
30 avg=amt/Db;
31
32 printf("\nAverage composition is %.2f kmol\n",avg);
33
34 L=4000;          //latent heat
35 R=2.1;
36
37 h=R*L;
38 printf("Heat required to produce reflux per kmol :
39 %d kJ\n",h);
40 //End

```

---

### Scilab code Exa 11.14 Ideal plates required

```

1 clear;
2clc;

```

```

3 xdo = 0.98; // per cent of ortho top
               product
4 xwo = 0.125; // per cent of ortho bottom
               product
5
6 function [f]=product(x)
7     f(1) = 100 - x(1) - x(2); //x(1) is D and x
                                (2) is W
8     f(2) = 60 - x(1)*xdo - x(2)*xwo;
9     funcprot(0);
10 endfunction
11 x = [0,0];
12 y = fsolve(x,product)
13 printf("\n D = %.2f kmol & W = %.2f kmol",y(1),y(2))
   ;
14
15 printf("\n Let us assume that the distillate
           contains 0.6 mole per cent meta and 1.4 mole per
           cent para");
16 printf("\n Component          Feed
           Distillate
           Bottoms      ");
17 printf("\n           (kmol)    (mole per cent)    (
           kmol)    (mole per cent)    (kmol)    (mole per cent
           )      ");
18 printf("\n Ortho   %.3f      %.2f      %.2f
           %.2f      %.2f      %.2f
           " ,60,60,y(1)*0.98,98,y(2)*0.125,12.5);
19 printf("\n Meta    %.3f      %.2f      %
           %.2f      %.2f      %
           %.2f      %.2f      "
           ,4,4,y(1)*0.006,0.6,y(2)*0.083,8.3);
20 printf("\n Para    %.3f      %.2f      %
           .2f      %.2f      %
           %.2f      %.2f
           " ,36,36,y(1)*0.014,1.4,y(2)*0.792,79.2);
21
22 ao = 1.7; // relative volatility of ortho
               relative to para

```

```

23 am = 1.16;           // relative volatility of meta
                        relative to para
24 ap =1;              // relative volatility of para
                        w.r.t. to itself
25 xso = 0.125;
26 xsm = 0.083;
27 xsp = 0.792;
28 xwo = 0.125;
29 xwp = 0.083;
30 xwm = 0.792;
31 yso = ao*xso/(ao*xso+ap*xsp+am*xsm);
32 ysm = am*xsm/(ao*xso+ap*xsp+am*xsm);
33 ysp = ap*xsp/(ao*xso+ap*xsp+am*xsm);
34 //Equations of operating lines
35 //Above the feed point
36 Ln = 5*y(1);        //Liquid downflow
37 Vn = 6*y(1);        //Vapour up
38 //Assuming the feed is liquid at its boiling point
39 F = 100;              //feed
40 Lm = Ln+F;            //liquid downflow
41Vm = Lm-y(2);          //Vapour up
42 x1o = poly([0], 'x1o');
43 x11 = roots(yso - (Lm/Vm)*x1o + (y(2)/Vm)*xwo);
44 x1p = poly([0], 'x1p');
45 x12 = roots(ysp - (Lm/Vm)*x1p + (y(2)/Vm)*xwp);
46 x1m = poly([0], 'x1m');
47 x13 = roots(ysm - (Lm/Vm)*x1m + (y(2)/Vm)*xwm);
48 x1 = [x11 x13 x12];
49 ax1 = [ao*x11 am*x13 ap*x12];
50 y1 = [ax1(1)/(ax1(1)+ax1(2)+ax1(3)) ax1(2)/(ax1(1) +
               ax1(2)+ax1(3)) ax1(3)/(ax1(1)+ax1(2)+ax1(3))];
51 x2o = poly([0], 'x2o');
52 x21 = roots(y1(1) - (Lm/Vm)*x2o + (y(2)/Vm)*xwo);
53 x2p = poly([0], 'x2p');
54 x22 = roots(y1(3) - (Lm/Vm)*x2p + (y(2)/Vm)*xwp);
55 x2m = poly([0], 'x2m');
56 x23 = roots(y1(2) - (Lm/Vm)*x2m + (y(2)/Vm)*xwm);
57 x2 = [x21 x23 x22];

```

```

58 printf("\n                  plate compositions below the
      feed plate");
59 printf("\n Component           xs
      axs          ys          x1
                  ax1          y1          x2");
60 printf("\n      o          %.3f          %.3f          %.3f          %
      .3f          %.3f          %.3f", xso , ao*xso , yso , x1(1)
      , ax1(1) , y1(1) , x2(1));
61 printf("\n      m          %.3f          %.3f          %.3f          %
      .3f          %.3f          %.3f", xsm , am*xsm , ysm , x1(2)
      , ax1(2) , y1(2) , x2(2));
62 printf("\n      p          %.3f          %.3f          %.3f          %
      .3f          %.3f          %.3f", xsp , ap*xsp , ysp , x1(3)
      , ax1(3) , y1(3) , x2(3));
63 printf("\n                  %.3f          %.3f          %.3f          %
      .3f          %.3f          %.3f", xso+xsm+xsp , ao*xso+
      am*xsm+ap*xsp , yso+ysm+ysp , x1(1)+x1(2)+x1(3) , ax1
      (1)+ax1(2)+ax1(3) , y1(1)+y1(2)+y1(3) , x2(1)+x2(2)-
      x2(3));
64
65 ax2 = [ao*x2(1) am*x2(2) ap*x2(3)];
66 y2 = [ax2(1)/(ax2(1)+ax2(2)+ax2(3)) ax2(2)/(ax2(1)-
      ax2(2)+ax2(3)) ax2(3)/(ax2(1)+ax2(2)+ax2(3))];
67 x3o = poly([0] , 'x3o');
68 x31 = roots(yso - (Lm/Vm)*x3o + (y(2)/Vm)*xwo);
69 x3p = poly([0] , 'x3p');
70 x32 = roots(ysp - (Lm/Vm)*x3p + (y(2)/Vm)*xwp);
71 x3m = poly([0] , 'x3m');
72 x33 = roots(ysm - (Lm/Vm)*x3m + (y(2)/Vm)*xwm);
73 x3 = [x31 x33 x32];
74
75 ax3 = [ao*x3(1) am*x3(2) ap*x3(3)];
76 y3 = [ax3(1)/(ax3(1)+ax3(2)+ax3(3)) ax3(2)/(ax3(1)-
      ax3(2)+ax3(3)) ax3(3)/(ax3(1)+ax3(2)+ax3(3))];

```

```

77 x4o = poly([0], 'x4o');
78 x41 = roots(yso - (Lm/Vm)*x4o + (y(2)/Vm)*xwo);
79 x4p = poly([0], 'x4p');
80 x42 = roots(ysp - (Lm/Vm)*x4p + (y(2)/Vm)*xwp);
81 x4m = poly([0], 'x4m');
82 x43 = roots(ysm - (Lm/Vm)*x4m + (y(2)/Vm)*xwm);
83 x4 = [x41 x43 x42];
84
85 ax4 = [ao*x4(1) am*x4(2) ap*x4(3)];
86 y4 = [ax4(1)/(ax4(1)+ax4(2)+ax4(3)) ax4(2)/(ax4(1)+  

     ax4(2)+ax4(3)) ax4(3)/(ax4(1)+ax4(2)+ax4(3))];
87 x5o = poly([0], 'x5o');
88 x51 = roots(yso - (Lm/Vm)*x5o + (y(2)/Vm)*xwo);
89 x5p = poly([0], 'x5p');
90 x52 = roots(ysp - (Lm/Vm)*x5p + (y(2)/Vm)*xwp);
91 x5m = poly([0], 'x5m');
92 x53 = roots(ysm - (Lm/Vm)*x5m + (y(2)/Vm)*xwm);
93 x5 = [x51 x53 x52];
94
95 ax5 = [ao*x5(1) am*x5(2) ap*x5(3)];
96 y5 = [ax5(1)/(ax5(1)+ax5(2)+ax5(3)) ax5(2)/(ax5(1)+  

     ax5(2)+ax5(3)) ax5(3)/(ax5(1)+ax5(2)+ax5(3))];
97 x6o = poly([0], 'x6o');
98 x61 = roots(yso - (Lm/Vm)*x6o + (y(2)/Vm)*xwo);
99 x6p = poly([0], 'x6p');
100 x62 = roots(ysp - (Lm/Vm)*x6p + (y(2)/Vm)*xwp);
101 x6m = poly([0], 'x6m');
102 x63 = roots(ysm - (Lm/Vm)*x6m + (y(2)/Vm)*xwm);
103 x6 = [x61 x63 x62];
104
105 ax6 = [ao*x6(1) am*x6(2) ap*x6(3)];
106 y6 = [ax6(1)/(ax6(1)+ax6(2)+ax6(3)) ax6(2)/(ax6(1)+  

     ax6(2)+ax6(3)) ax6(3)/(ax6(1)+ax6(2)+ax6(3))];
107 x7o = poly([0], 'x7o');
108 x71 = roots(yso - (Lm/Vm)*x7o + (y(2)/Vm)*xwo);
109 x7p = poly([0], 'x7p');
110 x72 = roots(ysp - (Lm/Vm)*x7p + (y(2)/Vm)*xwp);
111 x7m = poly([0], 'x7m');

```

```

112 x73 = roots(ysm - (Lm/Vm)*x7m + (y(2)/Vm)*xwm);
113 x7 = [x71 x73 x72];
114 printf("\n Component           ax2
           y2           x3           ax3
           y3           x4           ax4");
115 printf("\n   o           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,ax2(1),y2(1),x3(1),
           ax3(1),y3(1),x4(1),ax4(1));
116 printf("\n   m           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,xsm,am*xsm,ysm,x1(2)
           ,ax1(2),y1(2),x2(2));
117 printf("\n   p           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,xsp,ap*xsp,ysp,x1(3)
           ,ax1(3),y1(3),x2(3));
118
119 printf("\n Component           y4
           x5           ax5           y5
           x6           ax6           y6");
120 printf("\n   o           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,y4(1),x5(1),ax5(1),
           y5(1),x6(1),ax6(1),y6(1));
121 printf("\n   m           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,y4(2),x5(2),ax5(2),
           y5(2),x6(2),ax6(2),y6(2));
122 printf("\n   p           %.3f           %.3f           %.3f           %
           .3f           %.3f           %.3f           ,y4(3),x5(3),ax5(3),
           y5(3),x6(3),ax6(3),y6(3));
123
124
125 ax7 = [ao*x7(1) am*x7(2) ap*x7(3)];
126 y7 = [ax7(1)/(ax7(1)+ax7(2)+ax7(3)) ax7(2)/(ax7(1) +
           ax7(2)+ax7(3)) ax7(3)/(ax7(1)+ax7(2)+ax7(3))];
```

```

127 x8o = poly([0], 'x8o');
128 x81 = roots(yso - (Ln/Vn)*x8o + (y(2)/Vn)*xwo);
129 x8p = poly([0], 'x8p');
130 x82 = roots(ysp - (Ln/Vn)*x8p + (y(2)/Vn)*xwp);
131 x8m = poly([0], 'x8m');
132 x83 = roots(ysm - (Ln/Vn)*x8m + (y(2)/Vn)*xwm);
133 x8 = [x81 x83 x82];
134
135 ax8 = [ao*x8(1) am*x8(2) ap*x8(3)];
136 y8 = [ax8(1)/(ax8(1)+ax8(2)+ax8(3)) ax8(2)/(ax8(1) +
    ax8(2)+ax8(3)) ax8(3)/(ax8(1)+ax8(2)+ax8(3))];
137 x9o = poly([0], 'x9o');
138 x91 = roots(yso - (Ln/Vn)*x9o + (y(2)/Vn)*xwo);
139 x9p = poly([0], 'x9p');
140 x92 = roots(ysp - (Ln/Vn)*x9p + (y(2)/Vn)*xwp);
141 x9m = poly([0], 'x9m');
142 x93 = roots(ysm - (Ln/Vn)*x9m + (y(2)/Vn)*xwm);
143 x9 = [x91 x93 x92];
144
145 printf("\n Component           x7
          ax7      y7      x8
                  ax8      y8      x9");
146 printf("\n      o           %.3f
      %.3f      %.3f      %.3f      %
      .3f      %.3f      %.3f", x7(1), ax7(1), y7(1),
      x8(1), ax8(1), y8(1), x9(1));
147 printf("\n      m           %.3f
      %.3f      %.3f      %.3f      %
      .3f      %.3f      %.3f", x7(2), ax7(2), y7(2),
      x8(2), ax8(2), y8(2), x9(2));
148 printf("\n      p           %.3f
      %.3f      %.3f      %.3f      %
      .3f      %.3f      %.3f", x7(3), ax7(3), y7(3),
      x8(3), ax8(3), y8(3), x9(3));
149
150 printf("\n Component           x7
          ax7      y7      x8
                  ax8      y8      x9");

```

```

151 printf("\n      o          %.3f      %.3f      %.3f      %\n      .3f      %.3f      %.3f      %.3f      ,y4(1),x5(1),ax5(1),\n      y5(1),x6(1),ax6(1),y6(1));\n152 printf("\n      m          %.3f      %.3f      %.3f      %\n      .3f      %.3f      %.3f      %.3f      ,y4(2),x5(2),ax5(2),\n      y5(2),x6(2),ax6(2),y6(2));\n153 printf("\n      p          %.3f      %.3f      %.3f      %\n      .3f      %.3f      %.3f      %.3f      ,y4(3),x5(3),ax5(3),\n      y5(3),x6(3),ax6(3),y6(3));\n
```

---

### Scilab code Exa 11.15 Minimum reflux ratio

```

1 clear all;\n2 clc;\n3 printf("\n Example 11.15");\n4\n5 printf("\n Dew point calculation");\n6 xd1 = 0.975;           //n-C4 light distillate\n7 xd2 = 0.025;           //C5 heavy key distillate\n8 Td = 344;              //temperature in kelvins\n9 K1 = 1.05;             //Eqilibrium constant\n   calculation for n-C4 at 344 K\n10 K2 = 0.41;             //Equilibrium constant\n   calculation for C5 at 344K\n11 //By a dew point calculation\n12 //sum(xd)=sum(xd/K)\n13 printf("\n Component      xd          Td = 344\n      K");\n14 printf("\n      xd/K ");\n15 printf("\n n-C4          %.3f      %.2f\n      %.3f",xd1,K1,xd1/K1);\n
```

```

16  printf("\n C5          %.3 f      %.2 f
           %.3 f", xd2, K2, xd2/K2);
17  printf("\n          %.1 f
           %.3 f", xd1+xd2, (xd1/K1)+(xd2/K2));
18
19  K11 = 1.04;
20  K21 = 0.405;
21 //Calculation for xd at 343 K
22 x = poly([0], 'x');
23 x1 = roots(x/K11 + (1-x)/K21);
24 printf("\n

           Td = 343 K");
25 printf("\n

           K      xd/K");
26 printf("\n

           %.3 f    %.3 f", K11, x1/K11);
27 printf("\n

           %.3 f    %.3 f", K21, (1-x1)/K21);
28 printf("\n

           %.3 f", x1/K11+(1-x1)/K21);
29 printf("\n At 343 K      K1 = 1.04      K2 = 0.405
           from fig.11.39");
30
31 printf("\n\n\n Estimation of still temperature Ts");
32 //sum(xw) = sum(K*xw)
33 K31 = 3.05;           //equilibrium const at 419 K
34 K32 = 1.6;            //equilibrium const at 419 K
35 K33 = 0.87;           //equilibrium const at 419 K
36 K34 = 0.49;           //equilibrium const at 419 K
37 xw1 = 0.017;          //mass fraction of n-C4
38 xw2 = 0.367;          //mass fraction of C5
39 xw3 = 0.283;          //mass fraction of C6
40 xw4 = 0.333;          //mass fraction of C7

```

```

41
42 printf("\n At Ts = 416 K equilibrium constants are
        from fig.11.39");
43 printf("\n n-C4           C5           C6           C7");
44     ;
45 printf("\n   %.2f       %.2f       %.2f       %.2f"
        ,K31,K32,K33,K34);
46 printf("\n\n      at Ts = 416 K");
47 printf("\n n-C4           C5           C6           C7");
48     ;
49 printf("\n   %.3f       %.3f       %.3f       %.3f"
        ,xw1*K31,xw2*K32,xw3*K33,xw4*K34);
50 printf("\n Sum of Kxw = %d",xw1*K31+xw2*K32+xw3*K33+
        xw4*K34);
51 printf("\n Hence the still temperature Ts = 416 K");
52 printf("\n\n Calculation of feed condition");
53 printf("\n Component    xf           Tb = 377K
        Tb = 376 K");
54     ;
55 printf("\n\n K           Kxf");
56     ;
57 xf1 = 0.40;
58 xf2 = 0.23;
59 xf3 = 0.17;
60 xf4 = 0.20;
61 Kb1 = 1.80;           //equilibrium constants at
        377 K for n-C4
62 Kb2 = 0.81;           //equilibrium constants at
        377 K for C5
63 Kb3 = 0.39;           //equilibrium constants at
        377 K for C6
64 Kb4 = 0.19;           //equilibrium constants at
        377 K for C7
65 Kb11 = 1.78;          //equilibrium constants at
        377 K for n-C4
66 Kb21 = 0.79;          //equilibrium constants at
        377 K for C5
67 Kb31 = 0.38;          //equilibrium constants at
        377 K for C6

```

```

64 Kb41 = 0.185; // equilibrium constants at
                  377 K for C7
65 printf("\n n-C4      %.2f      %.2f      %.3f
          %.2f      %.3f", xf1, Kb1, xf1*Kb1, Kb11, xf1*Kb11);
66 printf("\n C5      %.2f      %.2f      %.3f
          %.2f      %.3f", xf2, Kb2, xf2*Kb2, Kb21, xf2*Kb21);
67 printf("\n C6      %.2f      %.2f      %.3f
          %.2f      %.3f", xf3, Kb3, xf3*Kb3, Kb31, xf3*Kb31);
68 printf("\n C7      %.2f      %.2f      %.3f
          %.2f      %.3f", xf4, Kb4, xf4*Kb4, Kb41, xf4*Kb41);
69 printf("\n
          %.3f", xf1*Kb1+xf2*Kb2+xf3*Kb3+xf4*Kb4
          , xf1*Kb11+xf2*Kb21+xf3*Kb31+xf4*Kb41);
70
71 // Calculation of pinch temperatures
72 printf("\n\n\n The upper pinch temperature ,Tn = %d K
          ", 343+0.33*(416-343));
73 printf("\n The lower pinch temperature ,Tm = %d K"
          , 343+0.67*(416-343));
74
75 // Calculation of approximate minimum reflux ratio .
76 printf("\n\n\n");
77 printf("\n Component      Tn = 367 K      Tm = 391 K
          xfh      axfh");
78 printf("\n
          "); a a
79 printf("\n n-C4      %.2f      %.2f
          ", 2.38, 2.00);
80 printf("\n C5      %.2f      %.2f
          ", 1.00, 1.00);
81 printf("\n C6      %.3f      %.3f
          %.2f      %.3f", 0.455, 0.464, 0.17, 0.077);
82 printf("\n C7      %.3f      %.3f
          %.2f      %.3f", 0.220, 0.254, 0.20, 0.044);
83 printf("\n
          %.3f", 0.077+0.044);
84 rf = xf1/xf2;

```

```

85 printf("\n rf = %.3f",rf);
86 xn4 = rf/[(1+rf)*(1+0.121)];
87 printf("\n xn4 = %.3f",xn4);
88 xn5 = xn4/rf;
89 printf("\n xn5 = %.3f",xn5);
90 Rm = [1/(2.38-1)]*(0.975/0.563)-2.38*(0.025/0.325);
91 printf("\n Rm = %.2f",Rm);
92
93 //The streams in the column
94 D = 40;
95 Ln = D*Rm;
96 Vn = Ln+D;
97 F = 100;
98 Lm = Ln + F;
99 W = 60;
100Vm = Lm - W;
101 Ratio = Lm/W;
102 printf("\n Ln = %.1f kmol",44.8);
103 printf("\n Vn = %.1f kmol",84.8);
104 printf("\n Lm = %.1f kmol",144.8);
105 printf("\n Vn = %.1f kmol",84.4);
106 printf("\n Lm/W = %.2f",Ratio);
107 //Check on minimum reflux ratio
108 //xn = xd/(a-1)Rm
109 xn = xd1/[(2.38-1)*Rm];
110 printf("\n For n-C4.....xn = %.3f",xn);
111 xn1 = 1-xn;
112 printf("\n For n-C5...xn = %.3f",xn1);
113 printf("\n Temperature check for upper pinch gives
           sum of K*xn = ");
114 sumKxn = 1.62*xn +0.68*xn1;
115 printf("%.3f",sumKxn);

```

---

### Scilab code Exa 11.16 Minimum reflux ratio

```

1 //Example 11.16
2
3 clear;
4 clc;
5
6 printf("\tExample 11.16\n");
7
8 //Mole fractions
9 xf=[0.40 0.35 0.25];
10 xd=[0.534 0.453 0.013];
11 xw=[0 0.04 0.96];
12
13 //Amount of feed , product , bottom in kmol
14 F=[40 35 25];
15 D=[40 34 1];
16 W=[0 1 24];
17
18 //roots of equation
19 theta=[1.15 1.17];
20
21 //relative volatility
22 alpha=[2.7 2.22 1];
23
24 //Underwoods 1st equation for q=1
25 sums=[0 0];
26 for i=1:2
27     for j=1:3
28         sums(i)=sums(i)+(alpha(j)*xf(j)/(alpha(j)-
29             theta(i)));
30     end
31 end
32 printf("\nFrom Underwoods 1st eq\n");
33 printf("The value of 1-q at theta = 1.15 and 1.17
34     are");
35 disp(sums);
36 sum2=0;
37
38 //Underwoods 2nd equation for minimum reflux ratio
39
40 %for i=1:2
41 %    for j=1:3
42 %        sum2=sum2+(alpha(j)*xf(j)/(alpha(j)-
43 %            theta(i)));
44 %    end
45 %end
46 %sum2
47
48 %for i=1:2
49 %    for j=1:3
50 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
51 %            theta(i)));
52 %    end
53 %end
54 %sum2
55
56 %for i=1:2
57 %    for j=1:3
58 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
59 %            theta(i)));
60 %    end
61 %end
62 %sum2
63
64 %for i=1:2
65 %    for j=1:3
66 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
67 %            theta(i)));
68 %    end
69 %end
70 %sum2
71
72 %for i=1:2
73 %    for j=1:3
74 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
75 %            theta(i)));
76 %    end
77 %end
78 %sum2
79
80 %for i=1:2
81 %    for j=1:3
82 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
83 %            theta(i)));
84 %    end
85 %end
86 %sum2
87
88 %for i=1:2
89 %    for j=1:3
90 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
91 %            theta(i)));
92 %    end
93 %end
94 %sum2
95
96 %for i=1:2
97 %    for j=1:3
98 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
99 %            theta(i)));
100 %    end
101 %end
102 %sum2
103
104 %for i=1:2
105 %    for j=1:3
106 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
107 %            theta(i)));
108 %    end
109 %end
110 %sum2
111
112 %for i=1:2
113 %    for j=1:3
114 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
115 %            theta(i)));
116 %    end
117 %end
118 %sum2
119
120 %for i=1:2
121 %    for j=1:3
122 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
123 %            theta(i)));
124 %    end
125 %end
126 %sum2
127
128 %for i=1:2
129 %    for j=1:3
130 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
131 %            theta(i)));
132 %    end
133 %end
134 %sum2
135
136 %for i=1:2
137 %    for j=1:3
138 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
139 %            theta(i)));
140 %    end
141 %end
142 %sum2
143
144 %for i=1:2
145 %    for j=1:3
146 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
147 %            theta(i)));
148 %    end
149 %end
150 %sum2
151
152 %for i=1:2
153 %    for j=1:3
154 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
155 %            theta(i)));
156 %    end
157 %end
158 %sum2
159
160 %for i=1:2
161 %    for j=1:3
162 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
163 %            theta(i)));
164 %    end
165 %end
166 %sum2
167
168 %for i=1:2
169 %    for j=1:3
170 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
171 %            theta(i)));
172 %    end
173 %end
174 %sum2
175
176 %for i=1:2
177 %    for j=1:3
178 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
179 %            theta(i)));
180 %    end
181 %end
182 %sum2
183
184 %for i=1:2
185 %    for j=1:3
186 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
187 %            theta(i)));
188 %    end
189 %end
190 %sum2
191
192 %for i=1:2
193 %    for j=1:3
194 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
195 %            theta(i)));
196 %    end
197 %end
198 %sum2
199
200 %for i=1:2
201 %    for j=1:3
202 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
203 %            theta(i)));
204 %    end
205 %end
206 %sum2
207
208 %for i=1:2
209 %    for j=1:3
210 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
211 %            theta(i)));
212 %    end
213 %end
214 %sum2
215
216 %for i=1:2
217 %    for j=1:3
218 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
219 %            theta(i)));
220 %    end
221 %end
222 %sum2
223
224 %for i=1:2
225 %    for j=1:3
226 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
227 %            theta(i)));
228 %    end
229 %end
230 %sum2
231
232 %for i=1:2
233 %    for j=1:3
234 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
235 %            theta(i)));
236 %    end
237 %end
238 %sum2
239
240 %for i=1:2
241 %    for j=1:3
242 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
243 %            theta(i)));
244 %    end
245 %end
246 %sum2
247
248 %for i=1:2
249 %    for j=1:3
250 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
251 %            theta(i)));
252 %    end
253 %end
254 %sum2
255
256 %for i=1:2
257 %    for j=1:3
258 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
259 %            theta(i)));
260 %    end
261 %end
262 %sum2
263
264 %for i=1:2
265 %    for j=1:3
266 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
267 %            theta(i)));
268 %    end
269 %end
270 %sum2
271
272 %for i=1:2
273 %    for j=1:3
274 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
275 %            theta(i)));
276 %    end
277 %end
278 %sum2
279
280 %for i=1:2
281 %    for j=1:3
282 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
283 %            theta(i)));
284 %    end
285 %end
286 %sum2
287
288 %for i=1:2
289 %    for j=1:3
290 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
291 %            theta(i)));
292 %    end
293 %end
294 %sum2
295
296 %for i=1:2
297 %    for j=1:3
298 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
299 %            theta(i)));
300 %    end
301 %end
302 %sum2
303
304 %for i=1:2
305 %    for j=1:3
306 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
307 %            theta(i)));
308 %    end
309 %end
310 %sum2
311
312 %for i=1:2
313 %    for j=1:3
314 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
315 %            theta(i)));
316 %    end
317 %end
318 %sum2
319
320 %for i=1:2
321 %    for j=1:3
322 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
323 %            theta(i)));
324 %    end
325 %end
326 %sum2
327
328 %for i=1:2
329 %    for j=1:3
330 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
331 %            theta(i)));
332 %    end
333 %end
334 %sum2
335
336 %for i=1:2
337 %    for j=1:3
338 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
339 %            theta(i)));
340 %    end
341 %end
342 %sum2
343
344 %for i=1:2
345 %    for j=1:3
346 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
347 %            theta(i)));
348 %    end
349 %end
350 %sum2
351
352 %for i=1:2
353 %    for j=1:3
354 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
355 %            theta(i)));
356 %    end
357 %end
358 %sum2
359
360 %for i=1:2
361 %    for j=1:3
362 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
363 %            theta(i)));
364 %    end
365 %end
366 %sum2
367
368 %for i=1:2
369 %    for j=1:3
370 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
371 %            theta(i)));
372 %    end
373 %end
374 %sum2
375
376 %for i=1:2
377 %    for j=1:3
378 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
379 %            theta(i)));
380 %    end
381 %end
382 %sum2
383
384 %for i=1:2
385 %    for j=1:3
386 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
387 %            theta(i)));
388 %    end
389 %end
390 %sum2
391
392 %for i=1:2
393 %    for j=1:3
394 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
395 %            theta(i)));
396 %    end
397 %end
398 %sum2
399
400 %for i=1:2
401 %    for j=1:3
402 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
403 %            theta(i)));
404 %    end
405 %end
406 %sum2
407
408 %for i=1:2
409 %    for j=1:3
410 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
411 %            theta(i)));
412 %    end
413 %end
414 %sum2
415
416 %for i=1:2
417 %    for j=1:3
418 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
419 %            theta(i)));
420 %    end
421 %end
422 %sum2
423
424 %for i=1:2
425 %    for j=1:3
426 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
427 %            theta(i)));
428 %    end
429 %end
430 %sum2
431
432 %for i=1:2
433 %    for j=1:3
434 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
435 %            theta(i)));
436 %    end
437 %end
438 %sum2
439
440 %for i=1:2
441 %    for j=1:3
442 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
443 %            theta(i)));
444 %    end
445 %end
446 %sum2
447
448 %for i=1:2
449 %    for j=1:3
450 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
451 %            theta(i)));
452 %    end
453 %end
454 %sum2
455
456 %for i=1:2
457 %    for j=1:3
458 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
459 %            theta(i)));
460 %    end
461 %end
462 %sum2
463
464 %for i=1:2
465 %    for j=1:3
466 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
467 %            theta(i)));
468 %    end
469 %end
470 %sum2
471
472 %for i=1:2
473 %    for j=1:3
474 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
475 %            theta(i)));
476 %    end
477 %end
478 %sum2
479
480 %for i=1:2
481 %    for j=1:3
482 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
483 %            theta(i)));
484 %    end
485 %end
486 %sum2
487
488 %for i=1:2
489 %    for j=1:3
490 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
491 %            theta(i)));
492 %    end
493 %end
494 %sum2
495
496 %for i=1:2
497 %    for j=1:3
498 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
499 %            theta(i)));
500 %    end
501 %end
502 %sum2
503
504 %for i=1:2
505 %    for j=1:3
506 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
507 %            theta(i)));
508 %    end
509 %end
510 %sum2
511
512 %for i=1:2
513 %    for j=1:3
514 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
515 %            theta(i)));
516 %    end
517 %end
518 %sum2
519
520 %for i=1:2
521 %    for j=1:3
522 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
523 %            theta(i)));
524 %    end
525 %end
526 %sum2
527
528 %for i=1:2
529 %    for j=1:3
530 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
531 %            theta(i)));
532 %    end
533 %end
534 %sum2
535
536 %for i=1:2
537 %    for j=1:3
538 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
539 %            theta(i)));
540 %    end
541 %end
542 %sum2
543
544 %for i=1:2
545 %    for j=1:3
546 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
547 %            theta(i)));
548 %    end
549 %end
550 %sum2
551
552 %for i=1:2
553 %    for j=1:3
554 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
555 %            theta(i)));
556 %    end
557 %end
558 %sum2
559
560 %for i=1:2
561 %    for j=1:3
562 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
563 %            theta(i)));
564 %    end
565 %end
566 %sum2
567
568 %for i=1:2
569 %    for j=1:3
570 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
571 %            theta(i)));
572 %    end
573 %end
574 %sum2
575
576 %for i=1:2
577 %    for j=1:3
578 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
579 %            theta(i)));
580 %    end
581 %end
582 %sum2
583
584 %for i=1:2
585 %    for j=1:3
586 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
587 %            theta(i)));
588 %    end
589 %end
590 %sum2
591
592 %for i=1:2
593 %    for j=1:3
594 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
595 %            theta(i)));
596 %    end
597 %end
598 %sum2
599
599
600 %for i=1:2
601 %    for j=1:3
602 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
603 %            theta(i)));
604 %    end
605 %end
606 %sum2
607
608 %for i=1:2
609 %    for j=1:3
610 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
611 %            theta(i)));
612 %    end
613 %end
614 %sum2
615
616 %for i=1:2
617 %    for j=1:3
618 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
619 %            theta(i)));
620 %    end
621 %end
622 %sum2
623
624 %for i=1:2
625 %    for j=1:3
626 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
627 %            theta(i)));
628 %    end
629 %end
630 %sum2
631
632 %for i=1:2
633 %    for j=1:3
634 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
635 %            theta(i)));
636 %    end
637 %end
638 %sum2
639
640 %for i=1:2
641 %    for j=1:3
642 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
643 %            theta(i)));
644 %    end
645 %end
646 %sum2
647
648 %for i=1:2
649 %    for j=1:3
650 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
651 %            theta(i)));
652 %    end
653 %end
654 %sum2
655
656 %for i=1:2
657 %    for j=1:3
658 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
659 %            theta(i)));
660 %    end
661 %end
662 %sum2
663
664 %for i=1:2
665 %    for j=1:3
666 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
667 %            theta(i)));
668 %    end
669 %end
670 %sum2
671
672 %for i=1:2
673 %    for j=1:3
674 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
675 %            theta(i)));
676 %    end
677 %end
678 %sum2
679
680 %for i=1:2
681 %    for j=1:3
682 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
683 %            theta(i)));
684 %    end
685 %end
686 %sum2
687
688 %for i=1:2
689 %    for j=1:3
690 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
691 %            theta(i)));
692 %    end
693 %end
694 %sum2
695
696 %for i=1:2
697 %    for j=1:3
698 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
699 %            theta(i)));
700 %    end
701 %end
702 %sum2
703
704 %for i=1:2
705 %    for j=1:3
706 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
707 %            theta(i)));
708 %    end
709 %end
710 %sum2
711
712 %for i=1:2
713 %    for j=1:3
714 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
715 %            theta(i)));
716 %    end
717 %end
718 %sum2
719
720 %for i=1:2
721 %    for j=1:3
722 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
723 %            theta(i)));
724 %    end
725 %end
726 %sum2
727
728 %for i=1:2
729 %    for j=1:3
730 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
731 %            theta(i)));
732 %    end
733 %end
734 %sum2
735
736 %for i=1:2
737 %    for j=1:3
738 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
739 %            theta(i)));
740 %    end
741 %end
742 %sum2
743
744 %for i=1:2
745 %    for j=1:3
746 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
747 %            theta(i)));
748 %    end
749 %end
750 %sum2
751
752 %for i=1:2
753 %    for j=1:3
754 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
755 %            theta(i)));
756 %    end
757 %end
758 %sum2
759
760 %for i=1:2
761 %    for j=1:3
762 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
763 %            theta(i)));
764 %    end
765 %end
766 %sum2
767
768 %for i=1:2
769 %    for j=1:3
770 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
771 %            theta(i)));
772 %    end
773 %end
774 %sum2
775
776 %for i=1:2
777 %    for j=1:3
778 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
779 %            theta(i)));
780 %    end
781 %end
782 %sum2
783
784 %for i=1:2
785 %    for j=1:3
786 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
787 %            theta(i)));
788 %    end
789 %end
790 %sum2
791
792 %for i=1:2
793 %    for j=1:3
794 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
795 %            theta(i)));
796 %    end
797 %end
798 %sum2
799
800 %for i=1:2
801 %    for j=1:3
802 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
803 %            theta(i)));
804 %    end
805 %end
806 %sum2
807
808 %for i=1:2
809 %    for j=1:3
810 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
811 %            theta(i)));
812 %    end
813 %end
814 %sum2
815
816 %for i=1:2
817 %    for j=1:3
818 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
819 %            theta(i)));
820 %    end
821 %end
822 %sum2
823
824 %for i=1:2
825 %    for j=1:3
826 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
827 %            theta(i)));
828 %    end
829 %end
830 %sum2
831
832 %for i=1:2
833 %    for j=1:3
834 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
835 %            theta(i)));
836 %    end
837 %end
838 %sum2
839
840 %for i=1:2
841 %    for j=1:3
842 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
843 %            theta(i)));
844 %    end
845 %end
846 %sum2
847
848 %for i=1:2
849 %    for j=1:3
850 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
851 %            theta(i)));
852 %    end
853 %end
854 %sum2
855
856 %for i=1:2
857 %    for j=1:3
858 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
859 %            theta(i)));
860 %    end
861 %end
862 %sum2
863
864 %for i=1:2
865 %    for j=1:3
866 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
867 %            theta(i)));
868 %    end
869 %end
860 %sum2
861
862 %for i=1:2
863 %    for j=1:3
864 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
865 %            theta(i)));
866 %    end
867 %end
868 %sum2
869
870 %for i=1:2
871 %    for j=1:3
872 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
873 %            theta(i)));
874 %    end
875 %end
876 %sum2
877
878 %for i=1:2
879 %    for j=1:3
880 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
881 %            theta(i)));
882 %    end
883 %end
884 %sum2
885
886 %for i=1:2
887 %    for j=1:3
888 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
889 %            theta(i)));
890 %    end
891 %end
892 %sum2
893
894 %for i=1:2
895 %    for j=1:3
896 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
897 %            theta(i)));
898 %    end
899 %end
900 %sum2
901
902 %for i=1:2
903 %    for j=1:3
904 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
905 %            theta(i)));
906 %    end
907 %end
908 %sum2
909
910 %for i=1:2
911 %    for j=1:3
912 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
913 %            theta(i)));
914 %    end
915 %end
916 %sum2
917
918 %for i=1:2
919 %    for j=1:3
920 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
921 %            theta(i)));
922 %    end
923 %end
924 %sum2
925
926 %for i=1:2
927 %    for j=1:3
928 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
929 %            theta(i)));
930 %    end
931 %end
932 %sum2
933
934 %for i=1:2
935 %    for j=1:3
936 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
937 %            theta(i)));
938 %    end
939 %end
940 %sum2
941
942 %for i=1:2
943 %    for j=1:3
944 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
945 %            theta(i)));
946 %    end
947 %end
948 %sum2
949
950 %for i=1:2
951 %    for j=1:3
952 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
953 %            theta(i)));
954 %    end
955 %end
956 %sum2
957
958 %for i=1:2
959 %    for j=1:3
960 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
961 %            theta(i)));
962 %    end
963 %end
964 %sum2
965
966 %for i=1:2
967 %    for j=1:3
968 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
969 %            theta(i)));
970 %    end
971 %end
972 %sum2
973
974 %for i=1:2
975 %    for j=1:3
976 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
977 %            theta(i)));
978 %    end
979 %end
980 %sum2
981
982 %for i=1:2
983 %    for j=1:3
984 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
985 %            theta(i)));
986 %    end
987 %end
988 %sum2
989
990 %for i=1:2
991 %    for j=1:3
992 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
993 %            theta(i)));
994 %    end
995 %end
996 %sum2
997
998 %for i=1:2
999 %    for j=1:3
1000 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1001 %            theta(i)));
1002 %    end
1003 %end
1004 %sum2
1005
1006 %for i=1:2
1007 %    for j=1:3
1008 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1009 %            theta(i)));
1010 %    end
1011 %end
1012 %sum2
1013
1014 %for i=1:2
1015 %    for j=1:3
1016 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1017 %            theta(i)));
1018 %    end
1019 %end
1020 %sum2
1021
1022 %for i=1:2
1023 %    for j=1:3
1024 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1025 %            theta(i)));
1026 %    end
1027 %end
1028 %sum2
1029
1030 %for i=1:2
1031 %    for j=1:3
1032 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1033 %            theta(i)));
1034 %    end
1035 %end
1036 %sum2
1037
1038 %for i=1:2
1039 %    for j=1:3
1040 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1041 %            theta(i)));
1042 %    end
1043 %end
1044 %sum2
1045
1046 %for i=1:2
1047 %    for j=1:3
1048 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1049 %            theta(i)));
1050 %    end
1051 %end
1052 %sum2
1053
1054 %for i=1:2
1055 %    for j=1:3
1056 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1057 %            theta(i)));
1058 %    end
1059 %end
1060 %sum2
1061
1062 %for i=1:2
1063 %    for j=1:3
1064 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1065 %            theta(i)));
1066 %    end
1067 %end
1068 %sum2
1069
1070 %for i=1:2
1071 %    for j=1:3
1072 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1073 %            theta(i)));
1074 %    end
1075 %end
1076 %sum2
1077
1078 %for i=1:2
1079 %    for j=1:3
1080 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1081 %            theta(i)));
1082 %    end
1083 %end
1084 %sum2
1085
1086 %for i=1:2
1087 %    for j=1:3
1088 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1089 %            theta(i)));
1090 %    end
1091 %end
1092 %sum2
1093
1094 %for i=1:2
1095 %    for j=1:3
1096 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1097 %            theta(i)));
1098 %    end
1099 %end
1100 %sum2
1101
1102 %for i=1:2
1103 %    for j=1:3
1104 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1105 %            theta(i)));
1106 %    end
1107 %end
1108 %sum2
1109
1110 %for i=1:2
1111 %    for j=1:3
1112 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1113 %            theta(i)));
1114 %    end
1115 %end
1116 %sum2
1117
1118 %for i=1:2
1119 %    for j=1:3
1120 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1121 %            theta(i)));
1122 %    end
1123 %end
1124 %sum2
1125
1126 %for i=1:2
1127 %    for j=1:3
1128 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1129 %            theta(i)));
1130 %    end
1131 %end
1132 %sum2
1133
1134 %for i=1:2
1135 %    for j=1:3
1136 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1137 %            theta(i)));
1138 %    end
1139 %end
1140 %sum2
1141
1142 %for i=1:2
1143 %    for j=1:3
1144 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1145 %            theta(i)));
1146 %    end
1147 %end
1148 %sum2
1149
1150 %for i=1:2
1151 %    for j=1:3
1152 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1153 %            theta(i)));
1154 %    end
1155 %end
1156 %sum2
1157
1158 %for i=1:2
1159 %    for j=1:3
1160 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1161 %            theta(i)));
1162 %    end
1163 %end
1164 %sum2
1165
1166 %for i=1:2
1167 %    for j=1:3
1168 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1169 %            theta(i)));
1170 %    end
1171 %end
1172 %sum2
1173
1174 %for i=1:2
1175 %    for j=1:3
1176 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1177 %            theta(i)));
1178 %    end
1179 %end
1180 %sum2
1181
1182 %for i=1:2
1183 %    for j=1:3
1184 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1185 %            theta(i)));
1186 %    end
1187 %end
1188 %sum2
1189
1190 %for i=1:2
1191 %    for j=1:3
1192 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1193 %            theta(i)));
1194 %    end
1195 %end
1196 %sum2
1197
1198 %for i=1:2
1199 %    for j=1:3
1200 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1201 %            theta(i)));
1202 %    end
1203 %end
1204 %sum2
1205
1206 %for i=1:2
1207 %    for j=1:3
1208 %        sum2=(alpha(j)*xf(j)/(alpha(j)-
1209 %            theta(i)));
1210 %    end
1211 %end
1212 %sum2
1213
1214 %for i=1:2
1215 %    for j=1:3
1216 %        sum2=(alpha(j)*xf(j)/(alpha(j)-

```

```

37 for l=1:3
38     sum2=sum2+(alpha(1)*xd(l)/(alpha(1)-theta(2)));
39 end
40
41 Rm=sum2-1;
42 printf("\nMinimum Reflux ratio is %.3f\n",Rm);
43
44 //End

```

---

**Scilab code Exa 11.17** Number of theoretical plates required

```

1 //Example 11.17 Fenske 's Equation
2
3 clear;
4 clc;
5
6 printf("\tExample 11.17\n");
7
8 //From previous question data
9 xA_d=0.453;
10 xB_d=0.013;
11
12 xA_s=0.04;
13 xB_s=0.96;
14
15 alpha_av=2.22;
16
17 //By Fenske Equation for no. of plates
18
19 n=((log(xA_d*xB_s/(xA_s*xB_d)))/log(alpha_av))-1;
20
21 printf("\nMinimum no. of plates are %f or %d\n",n,n)
22 ;
23 //End

```

---

### Scilab code Exa 11.18 Xchange in n with R

```
1 //Example 11.18
2
3 clear;
4 clc;
5
6 printf("\tExample 11.18\n");
7
8 R=[1 2 5 10];
9 Rm=0.83;
10
11 nm=8.5-1;
12
13 //X are points on X-axis of graph
14 for i=1:4
15     X(i)=(R(i)-Rm)/(R(i)+1);
16 end
17
18 //Values at Y-axis for corresponding values of X-
19 //axis in graph are given by
20 Y=[0.55 0.32 0.15 0.08];
21
22 // where Y=(n+1)-(nm+1)/(n+2)
23
24 for i=1:4
25     n=poly([0], 'x');
26     N(i)=roots(((n+1)-(nm+1))-(Y(i)*(n+2)));
27 end
28
29 printf("\nThe values of R and n are\n");
30 for i=1:4
31     printf("\n\t%d \t %.2f", R(i), N(i));
```

```

32 end
33
34 printf("\n\nThe change in R and n can be seen as
      above\n");
35
36 //End

```

---

### Scilab code Exa 11.19 Optimum reflux ratio

```

1 clear all;
2 clc;
3 printf("\n Example 11.19");
4 //Data from fig. 11.42
5 a = [0 0.02 0.04 0.06 0.08 0.1 0.2 0.4 0.6 0.8 1.0];
6 b = [0.75 0.62 0.60 0.57 0.55 0.52 0.45 0.30 0.18
      0.09 0];
7 //a = (R-Rm)/(R+1)
8 //b = [(n+1)-(nm+1)]/(n+2)
9 R = [0.92 1.08 1.25 1.75 2.5 3.5 5.0 7.0 9.0];
10 n = [28.6 22.8 16.9 13.5 11.7 10.5 9.8 9.2 8.95];
11 plot(n,R);
12 xtitle("Plot of R vs n","n","R");
13 printf("\n Derivative calculated from the graph");
14 d = [110.0 34.9 9.8 3.8 1.7 0.6 0.4 0.2 0.05];
15 i=1;
16 while i <=9
17     s = R(i)+1 - (n(i)+7.72)/d(i);
18     if s <=0.0001 then
19         Ropt = R(i);
20         printf("\n Ropt = %.2f",Ropt);
21         break;
22     end
23     i=i+1;
24 end
25 printf("\n R is approximately %.1f percent of the

```

```
minimum reflux condition" ,1.25/0.86666666*100);
```

---

**Scilab code Exa 11.20** Plate efficiency for the given data

```
1 //Example 11.20
2
3 clear;
4 clc;
5
6 printf("\tExample 11.20\n");
7
8 //Mole fraction
9 xf=[0.2 0.3 0.2 0.3];
10
11 //Viscosity at mean tower temp. in mNs/m^2
12 uL=[0.048 0.112 0.145 0.188];
13
14 //Viscosity of water in mNs/m^2
15 uw=1;
16
17 sums=0;
18 for i=1:4
19     sums=sums+(xf(i)*uL(i)/uw);
20 end
21
22 //Efficiency by DRICKAMER and BRADFORD
23 E=0.17-(0.616*log10(sums));
24
25 printf("\nEfficiency is %.2f",E);
26
27 //End
```

---

# Chapter 12

## Absorption of gases

Scilab code Exa 12.1 Overall liquid film coefficient

```
1 clear;
2 clc;
3 printf("\n Example 12.1");
4 //Overall liquid transfer coefficient KLa = 0.003
   kmol/s.m^3(kmol/m^3)
5
6 //(1/KLa)=(1/kLa)+(1/HkGa)
7 // let (KLa)=x
8 x = 0.003;
9 overall = 1/x;
10
11 //For the absorption of a moderately soluble gas it
   is reasonable to assume that the liquid and gas
   phase resistances are of the same order
   ofmagnitude , assuming them to be equal.
12 //(1/KLa)=(1/kLa)+(1/HkGa)
13 //let 1/kLa = 1/HkGa = y
14 y = (1/(2*x));
15 z = (1/y);           //z is in kmol/s m^3(kmol/m
   ^3)
16 printf("\n For S02:");
```

```

17 printf("\n      kGa = %f kmol/s m^3(kN/m^2)",z/50);
18 printf("\n For NH3:");
19 d_SO2 = 0.103;           // diffusivity at 273K for
                           SO2 in cm^2/sec
20 d_NH3 = 0.170;          // diffusivity at 273K for
                           NH3 in cm^2/sec
21 printf("\n      kGa = %f kmol/s m^3(kN/m^2)",(z/50)*(d_NH3/d_SO2)^0.56);
22 printf("\n For a very soluble gas such as NH3, kGa = KGa.");
23 printf("\n For NH3 the liquid-film resistance will
                           be small, and:");
24 printf("\n      kGa =KGa = %f kmol/s m^3(kN/m^2)"
                           ,(z/50)*(d_NH3/d_SO2)^0.56);

```

---

### Scilab code Exa 12.2 Mass transfer coefficient

```

1 clear;
2 clc;
3 printf("\n Example 12.2");
4 G = 2;                      // air flow rate
                               in kg/m^2.sec
5 Re = 5160;                   //Re stands for
                               Reynolds number
6 f = 0.02;                    // friction factor
                               = R/pu^2
7 d_SO2 = 0.116*10^(-4);      // diffusion
                               coefficient in m^2/sec
8 v = 1.8*10^(-5);            // viscosity
                               in mNs/m^2
9 p = 1.154;                   // density is in kg
                               /m^3
10
11 // (hd/u)(Pm/P)(u/p/d_SO2)^0.56 = BRe^(-0.17)=jd
12

```

```

13 function[x]=a1()
14     x = (v/(p*d_S02))^(0.56);
15     funcprot(0);
16 endfunction
17
18 //jd = f =R/pu^2
19 function[y]=a2()
20     y = f/a1();
21     funcprot(0);
22 endfunction
23 //G = pu
24 u = (G/p);                                //u is in m/sec
25
26 function[x1]= a3()
27     x1 = a2()*u;
28     funcprot(0);
29 endfunction
30
31 function[d]=d1()
32     d = Re*v/(G);
33     funcprot(0);
34 endfunction
35
36 printf("\n d = %f mm",d1());
37 R = 8314;                                     //R is in m^3(N/m^2)/K kmol
38 T = 298;                                       //T is in Kelvins
39 function[kG]=kG1()
40     kG = a3()/(R*T);
41     funcprot(0);
42 endfunction
43
44 printf("\n kG = %.2f *10^(-8) kmol/m^2 sec (N/m^2)",kG1
    ()*10^(8));
45 printf("\n kG = %.2f *10^(-4) kg SO2/m^2 sec (kN/m^2)", 
    kG1()*10^(7)*64);

```

---

### Scilab code Exa 12.3 Overall transfer units required

```
1 clear;
2 clc;
3 printf("\n Example 12.3");
4 //as the system are dilute mole fractions are
    //approximately equal to mole ratios.
5 //At the bottom of the tower
6 y1 = 0.015;           //mole fraction
7 G = 1;                //Gas flow rate is in kg/m^2
    sec
8 //At the top of the tower
9 y2 = 0.00015;         //mole fraction
10 x2 = 0;
11 L = 1.6;              //liquid flow rate is in kg/m
    ^2.sec
12 Lm = 1.6/18;          //liquid flow rate is in kmol/m
    ^2.sec
13 Gm = 1.0/29;          //gas flow rate is in kmol/m^2.
    sec
14 x1 = poly([0], 'x1');
15 x11 = roots(Gm*(y1-y2)-Lm*(x1));
16 printf("\n     x1 = %f", x11);
17 function[ye1]=henry_law(x)
18     ye1 = 1.75*x;
19     funcprot(0);
20 endfunction
21 bottom_driving_force = y1 - henry_law(x11);
22 function[lm]=log_mean()
23     lm = (bottom_driving_force-y2)/log(
        bottom_driving_force/y2);
24     funcprot(0);
25 endfunction
26 NoG = (y1-y2)/log_mean();
```

```
27 NoL = NoG*1.75*(Gm/Lm);  
28 printf("\n      NoL =%.2f",NoL);
```

---

**Scilab code Exa 12.4** Height of transfer units and number of transfer units

```
1 clear;  
2 clc;  
3 printf("\n Example 12.4");  
4 Gm = 0.015;  
5 KGa = 0.04;  
6 top = 0.0003;  
7 Y1 = 0.03;  
8 Ye = 0.026;  
9 bottom = (Y1-Ye);  
10 log_mean_driving_force = (bottom-top)/log(bottom/top)  
    );  
11 Z = poly([0], 'Z');  
12 Z1 = roots(Gm*(Y1-top)-KGa*log_mean_driving_force*Z)  
    ;  
13 printf("\n      Z =%.2fm",Z1);  
14 HoG = Gm/KGa;  
15 printf("\n      Height of transfer unit HoG = %.3fm",HoG)  
    );  
16 printf("\n      Number of transfer units NoG = %f",ceil  
    (7.79/0.375));
```

---

**Scilab code Exa 12.5** Height of the tower

```
1 clear;  
2 clc;  
3 printf("\n Example 12.5");  
4 y1 = 0.10;  
5 Y1 = 0.10/(1-0.10);
```

```

6 y2 = 0.001;
7 Y2 = y2;
8 mass_flowrate_gas = 0.95;           //mass flow rate in
                                         kg/m^2.sec
9 mass_percent_air = (0.9*29/(0.1*17+0.9*29))*100;
10 mass_flowrate_air = (mass_percent_air*
                           mass_flowrate_gas); //in kg/m^2.sec
11 Gm = (mass_flowrate_air/29);
12 Lm = 0.65/18;                  //Lm is in kmol/m^2.sec
13 //A mass balance between a plane in the tower where
      the compositions are X and Y and the top of the
      tower gives:
14
15 //Y = 1.173X+0.001
16 X = 0:0.001:0.159;
17 Y = 1.173*X+0.001;
18 plot(X,Y);
19
20 x=[0.021 0.031 0.042 0.053 0.079 0.106 0.159];
21
22 PG = [12 18.2 24.9 31.7 50.0 69.6 114.0];
23 i=1;
24 while i<8
25     Y1(i) = PG(i)/(760-PG(i));
26     i=i+1;
27 end
28 plot(x,Y1);
29 // xlabel("Area under the curve is 3.82m","kmolNH3/
                                         kmolH2O","kmolNH3/kmol air");
30 xtitle("Operating and equilibrium lines","kmol NH3/
                                         kmol H2O","kmol NH3/kmol air");
31 //If the equilibrium line is assumed to be straight ,
      then:
32 //Gm(Y2      Y1) = KGaZdeltaPlm
33
34 //Top driving force
35 deltaY2 = 0.022;
36 //Bottom driving force

```

```

37 deltaY1 = 0.001;
38 deltaY1m = 0.0068;
39 Z = (0.0307*0.11)/(0.001*0.688);
40 printf("The height of the tower is %.2f meters",Z);

```

---

### Scilab code Exa 12.6 specific steam consumption

```

1 clear;
2 clc;
3 printf("\n Example 12.6");
4 printf("\n Number of theoretical plates = %d"
      ,(30*0.3));
5
6 //At the bottom of the tower:
7 //Flowrate of steam = Gm (kmol/m^2.s)
8 //Mole ratio of pentane in steam = Y1, and
9 //Mole ratio of pentane in oil = X1
10 X1 = 0.001;
11
12 //At the top of the tower:
13 //exit steam composition = Y2
14 //inlet oil composition = X2
15 X2 = 0.06;
16 //flowrate of oil = Lm (kmol/m^2.sec)
17
18 //The minimum steam consumption occurs when the exit
   steam stream is in equilibrium with the inlet
   oil, that is when:
19 //The equilibrium relation for the system may be
   taken as Ye = 3.0X, where Ye and X are expressed
   in mole ratios of pentane in the gas and liquid
   phases respectively.
20 Ye2 = X2*3;
21 //Lmin(X2      X1) = Gmin(Y2      Y1)
22 //If Y1 = 0, that is the inlet steam is pentane-free

```

```

        , then:
23 ratio_min = (X2 - X1)/Ye2;
24
25
26 //The operating line may be fixed by trial and error
   as it passes through the point (0.001, 0), and 9
   theoretical plates are required for the
   separation. Thus it is a matter of selecting the
   operating line which, with 9 steps, will give X2
   = 0.001 when X1 = 0.06. This is tedious but
   possible, and the problem may be better solved
   analytically since the equilibrium line is
   straight.
27
28 // let x = 1/A
29 //for a stripping operation
30 LHS = (X2-X1)/(X2);
31 printf("\n LHS = %f",LHS);
32 x = poly([0], 'x');
33 x1 = roots((x^10-x)-LHS*(x^(10)-1));
34 printf("\n 1/A = %.2 f",x1(9));
35 //A = Lm/mGm
36 printf("\n Gm/Lm = %.3 f",x1(9)/3);
37 printf("\n Actual/minimum Gm/Lm = %.2 f",0.457/0.328)
   ;
38
39 // If (actual Gm/Lm) / (min Gm/Lm) = 2,
40 printf("\n Actual Gm/Lm = %.3 f",.328*2);
41 x2 = 3*(0.656);
42 printf("\n 1/A = mGm/Lm = %.3 f",3*0.656);
43
44 //y = (1.968)^(N+1)
45 y = poly([0], 'y');
46 y1 = roots(0.983*(y-1)-(y-1.968));
47 N = log(y1)/log(1.968)-1;
48 printf("\n The actual number of plates are : %.1 f",
   ceil(N/0.3));

```

---

# Chapter 13

## Liquid liquid extraction

Scilab code Exa 13.1 Composition of final raffinate

```
1 clear all;
2 clc;
3 printf("\n Example 13.1");
4 printf("\n      (a) Countercurrent operation");
5 // (a) Countercurrent operation
6 S = 1.6*10^(-4);                      // Solvent flow rate
   in m^3/sec
7 mass_flowrate = (S*800);                // mass flow rate is
   in kg/sec
8
9 // Considering the solution , 400cm3/s = 4      10      4 m
   ^3/sec containing , say , a m^3/sec A and (5      10
   4      a) m3/sec B.
10 //Thus mass flow rate of A = 1200a kg/sec
11 //and mass flow rate of B = (4*10^(-4)-a)*1000 =
   (0.4-1000a)kg/sec
12 //a total of:                           (0.4+200a) kg/sec
13 C = poly([0], 'C');
14 C1 = roots(0.1*(0.4+200*C)-1200*C);
15 printf("\n Concentration of the solution is %.2f
   *10^(-5) m^3/sec", C1*10^5);
```

```

16 printf("\n mass flow rate of A = %.3f kg/sec" ,1200*
C1);
17 printf("\n mass flow rate of B =%.3f kg/sec"
,0.4+200*C1);
18 printf("\n Ratio of A/B in the feed , Xf = %.3f kg/kg
" ,0.041/0.366);
19
20 X = [0.05 0.10 0.15];
21 Y = [0.069 0.159 0.258];
22 plot(X,Y);
23 xtitle("Equilibrium curve","kg A/kg B","kg A/kg S");
24 //From The curve:
25 slope = 0.366/0.128;
26 printf("\n Slope of the equilibrium line is %.2f",
slope);
27
28 //Since pure solvent is added , Yn+1 = Y4 = 0 and a
line of slope 2.86 is drawn in such that stepping
off from Xf = 0.112 kg/kg to Y4 = 0 gives
exactly three stages. When Y4 = 0 , Xn = X3 =
0.057 kg/kg
29 printf("\n The composition of final raffinate is
0.057kg A/kg B");
30
31 printf("\n\n\n (b) Multiple contact");
32 printf("\n Stage 1");
33 printf("\n In this case %.4f kg/sec" ,0.128/3);
34 //and from the equilibrium curve , the extract
contains 0.18 A/kg S and (0.18 0.0427) =
0.0077 kg/s A.
35 printf("\n Thus raffinate from stage 1 contains %.4f
kg/sec Aand %.3f kg/sec B ,(0.041-0.0077) ,0.366)
;
36 X1 = 0.0333/0.366;
37 printf("\n X1 = %.3f kg/kg" ,0.0333/0.366);
38
39 printf("\n Stage 2");
40 //the extract contains 0.14 kg A/kg S

```

```

41 printf("\n The extract contains %.4f kg/sec A"
        ,0.14*0.0427);
42 //Thus: the raffinate from stage 2 contains (0.0333
        0.0060) = 0.0273 kg/s A and 0.366 kg/s B
43 X2 = (0.0273/0.366);
44 printf("\n X2 = %.3f kg/kg",X2);
45
46 printf("\n Stage 3");
47 //the extract contains 0.114 kg A/kg S
48 printf("\n The extract contains %.4f kg/sec A/kg S"
        ,0.114*0.0427);
49 printf("\n Thus the raffinate contains %.4f kg/sec A
        and %.3f kg/sec B", (0.0273-0.0049),0.366);
50 printf("\n The composition of final raffinate = %3f
        kg A/kg B",0.0224/0.366);

```

---

### Scilab code Exa 13.3 Overall transfer coefficient

```

1 clear all;
2 clc;
3 printf("\n Example 13.3");
4 //From the equilibrium relationship
5 CB1 = (0.0247*0.685);
6 printf("\n CB1* = %.4f kmol/m^3",CB1);
7 CB2 = (0.0247*0.690);
8 printf("\n CB2* = %.4f kmol/m^3",CB2);
9
10 //Thus the driving force at the bottom:
11 deltaC1 = (0.0169-0.0040);
12 printf("\n deltaC1 =%.4f kmol/m^3",deltaC1);
13 //Driving force at the top
14 deltaC2 = (0.0170-0.0115);
15 printf("\n deltaC2 = %.4f kmol/m^3",deltaC2);
16 function[x]= log_mean_driving_force()
17     x = (deltaC1 - deltaC2)/log((deltaC1)/deltaC2);

```

```

18     funcprot(0);
19 endfunction
20 printf("\n log mean driving force is given by
    deltaClm = %.4f kmol/m^3", log_mean_driving_force
());
21 KBa = (4.125*10^(-8))/(log_mean_driving_force()
    *0.0063);
22 printf("\n KBa = %.1f*10^(-4) kmol/sec m^3(kmol/m^3)
    ", KBa*10^4);
23 HoB = (1.27*10^(-3))/KBa;
24 printf("\n HoB = %.2f meters", HoB);

```

---

### Scilab code Exa 13.4 Surface mean droplet size

```

1 clear all;
2 clc;
3 printf("\n Example 13.4");
4 diameter = [2 3 4 5 6];
5 number = [30 120 200 80 20];
6
7 function[x] = Sum_d1cube()
8     sum = 0;
9     i = 1;
10    while (i <= 5)
11        sum = sum + number(i)*(diameter(i))^3;
12        i = i+1;
13    end
14    x = sum;
15    funcprot(0);
16 endfunction
17
18 function[y]=sum_d1square()
19     sum1 = 0;
20     j=1;
21     while(j<=5)

```

```

22         sum1 = sum1 + number(j)*(diameter(j))^2;
23         j= j+1;
24     end
25     y = sum1;
26     funcprot(0);
27 endfunction
28
29 function[z]= ds()
30     z = Sum_d1cube()/sum_d1square();
31     funcprot(0);
32 endfunction
33 printf("\n Mean droplet size = %.2f mm",ds());

```

---

**Scilab code Exa 13.5** Number of overall transfer units in raffinate phase

```

1 clear all;
2 clc;
3 printf("\n Example 13.5");
4 CSA = (%pi/4)*(0.075)^2; // cross
    sectional area is in m^2
5 V = (0.0044*3); // volume of
    packing is in m^3
6 C = 0.01; // concentration
    is in kg/kg
7 printf("\n mass of acid transferred to the ether %.4
    f kg/m^2.sec or %f kg/sec",0.05*(0.01-0)
    ,0.005*0.0044);
8 printf("\n Acid in the aqueous feed = %.2f kg/m^3.
    sec",0.25*0.04);
9 printf("\n Acid in the raffinate = %.3f kg/m^2.sec"
    ,0.01-0.005);
10 printf("\n Concentration of acid in the raffinate =
    %.2f kg/kg",0.005/0.25);
11 printf("\n At the top of the column");
12 CR2 = 0.040; // Concentration is in kg/

```

```

    kg
13 CR22 = 0.040*0.3;           // Concentration is in kg/
    kg
14 deltaC2 = (0.012-0.010);
15 printf("\n deltaC2 = %.3f kg/kg",deltaC2);
16 printf("\n\n At the bottom of the column");
17 CR1 = 0.20;                 // Concentration is in kg/
    kg
18 CR11 = 0.020*0.3          // Concentration is in kg/
    kg
19 deltaC1 = 0.006 -0;        // Concentration is in kg/
    kg
20 printf("\n deltaC1 = %.3f kg/kg",deltaC1);
21 deltaCRlm = (0.006-0.002)/log(0.006/0.002);
22 printf("\n Logarithmic driving force is : %.4f kg/kg
    ",deltaCRlm);
23 KRa = 0.000022/(0.01333*deltaCRlm);
24 printf("\n KRa = %.3f kg/m^3.sec(kg/kg)",KRa);
25 printf("\n Height of an overall transfer unit = %.2f
    m",0.25/KRa);
26 printf("\n The number of overall transfer units = %
    .2f ",3/0.54);

```

---

# Chapter 14

## Evaporation

**Scilab code Exa 14.1** Heat surface required

```
1 clear;
2 clc;
3 printf("\n Example 14.1");
4 //Assuming that the steam is dry and saturated at
   205 kN/m2, then from the Steam Tables in the
   Appendix, the steam temperature = 394 K at which
   the total enthalpy = 2530 kJ/kg.
5
6 //At 13.5 kN/m2, water boils at 325 K and, in the
   absence of data on the boiling point elevation,
   this will be taken as the temperature of
   evaporation, assuming an aqueous solution. The
   total enthalpy of steam at 325 K is 2594 kJ/kg.
7
8 //Thus the feed, containing 10 per cent solids, has
   to be heated from 294 to 325 K at which
   temperature the evaporation takes place.
9
10 printf("\n mass of dry solids = %.1f kg/sec"
       ,(7*10/100));
11 x = poly([0], 'x');
```

```

12 x1 = roots(0.7*100-50*(0.7+x));
13 printf("\n x = %.1f kg/sec",x1);
14 printf("\n Water to be evaporated = %.1f kg/sec"
15 , (7-0.7)-0.7);
15 printf("\n Summarising");
16 printf("\n Stream Solids Liquid
17 Total ");
17 printf("\n (kg/s) (kg/s)");
18 printf("\n Feed %.1f %.1f
19 %.1f ,x1 ,7-x1 ,x1+7-x1);
19 printf("\n Product %.1f %.1f
19 ,x1 ,x1 ,x1+x1);
20 printf("\n Evaporation %.1f
20 ,7-x1-x1 ,7-2*x1);
21 //Using a datum of 273K
22 q_entering = (7*3.76)*(294-273);
23 printf("\n Heat entering with the feed = %.1f kW",
23 q_entering);
24 q_leaving = (1.4*3.14)*(325-273);
25 printf("\n Heat leaving with the product = %.1f kW",
25 q_leaving);
26 printf("\n Heat leaving with the evaporated water =
26 %d kW",5.6*2594);
27 printf("\n Heat transferred from the steam = %d kW"
27 ,14526+228.6-552.7);
28 printf("\n The enthalpy of the condensed steam
28 leaving at 352.7 K = %.1f kJ/kg "
28 ,4.18*(352.7-273));
29 printf("\n The heat transferred from 1 kg steam = %
29 .1f kJ/kg",2530-333.2);
30 printf("\n Steam required = %.2f kg/s "
30 ,14202/2196.8);
31
32 //s the preheating of the solution and the sub-
32 cooling of the condensate represent but a small
32 proportion of the heat load, the temperature
32 driving force may be taken as the difference

```

between the temperatures of the condensing steam  
and the evaporating water , or:

```
33 printf("\n deltaT = %d deg K ",394-325);  
34 printf("\n Heat transfer area ,A = %.1f m^2"  
       ,14202/(3*69));
```

---

### Scilab code Exa 14.2A Forward feed

```
1 clear;  
2 clc;  
3 printf("\n Example 14.2 a");  
4 //Temperature of dry saturated steam at 205 kN/m2 =  
   394 K.  
5 //At a pressure of 13 kN/m2 (0.13 bar) , the boiling  
   point of water is 325 K, so that the  
   total temperature difference &T = (394 - 325) =  
   69deg K.  
6  
7  
8 //First Approximation.  
9  
10 //Assuming that: U1*deltaT1 = U2*deltaT2 = U3*  
    deltaT3  
11 //then substituting the values of U1, U2 and U3 and  
   &T = 69 deg K gives:  
12 printf("\n deltaT1 = %f deg K",13);  
13 printf("\n deltaT2 = %f deg K",20);  
14 printf("\n deltaT2 = %f degK",36);  
15  
16 //Since the feed is cold , it will be necessary to  
   have a greater value of T1 than given by this  
   analysis. It will be assumed that T1 = 18 deg K,  
   T2 = 17 deg K, T3 = 34 deg K.  
17  
18 //For steam to 1: T0 = 394 K and h0 = 2200 kJ/kg
```

```

19 //For steam to 2: T1 = 376 K and 1 = 2249 kJ/kg
20 //For steam to 3: T2 = 359 K and 2 = 2293 kJ/kg
21 // T3 = 325 K and 3 = 2377 kJ/kg
22
23 //Assuming that the condensate leaves at the steam
   temperature , then heat balances across each
   effect may be made as follows:
24
25 //Effect 1 :
26 //2200 D0 = 4 4.18(376 - 294) + 2249 D1
27
28 //Effect 2:
29 //2249 D1 + (4 - D1) 4.18(376 - 359) = 2293 D2
30
31 //Effect 3:
32 //2293 D2 + (4 - D1 - D2) 4.18(359 - 325) =
   2377 D3
33
34 printf("\n      Solids          liquor
           Total    ");
35 printf("\n      (kg/s)          (kg/s)
           (kg/s)    ");
36 printf("\nFeed      %.1f          %.1f
           %.1f    ",0.4,3.6,4.0);
37 printf("\nProduct    %.1f          %.1f
           %.1f    ",0.4,0.4,0.8);
38 printf("\nEvaporation          %.1f
           %.1f    ",3.6-0.4,4.0-0.8);
39
40 //D1 + D2 + D3 = 3.2 kg/s
41 D = [2177.94 -2293 0;-4.18*(359-325)
      2293-4.18*(359-325) -2377;1 1 1];
42 C = [-4*4.18*(376-359);-4*4.18*(359-325);3.2];
43 D1 = inv(D)*C;
44 printf("\n ans = %.3f kg/s",D1);
45
46 D0 = (4*4.18*(376-294)+2249*D1(1))/2200;
47 A1 = D0*2200/(3.1*18);

```

```

48 printf("\n A1 = %.1f m^2",A1);
49 A2 = D1(1)*2249/(2*17);
50 printf("\n A2 = %.1f m^2",A2);
51 A3 = D1(2)*2293/(1.1*34);
52 printf("\n A3 = %.1f m^2",A3);
53 printf("\n These three calculated areas are
           approximately equal, so that the temperature
           differences assumed may be taken as nearly
           correct");
54
55 //let , deltaT1 = x
56 //deltaT2 = y
57 //deltaT3 = z
58 X = [3.1 -2.0 0;0 2.0 -1.1;1 1 1];
59 A = [0;0;69];
60 T = inv(X)*A;
61 printf("\n deltaT1 = %d deg K",T(1));
62 printf("\n deltaT2 = %d deg K",T(2));
63 printf("\n deltaT3 = %d deg K",T(3));
64
65 //gghting the temperature differences to allow for
       the fact that the feed enters at ambient
       temperature gives:
66 //deltaT1 = 18 deg Kdelta T2 = 18 degK delta T3 = 33
       deg K
67 Steam_temp = 394;
68 T1 = Steam_temp-18;
69 printf("\n T1 = %d K",T1);
70 T2 = T1 - 18;
71 printf("\n T2 = %d K",T2);
72 T3 = T2 - 33;
73 printf("\n T3 = %d K",T3);
74
75 //The total evaporation (D1 + D2 + D3) is obtained
       from a material balance:
76 printf("\n           Solids          liquor
                  Total      ");
77 printf("\n           (kg/s)        (kg/s)

```

```

    (kg/s) " );
78 printf("\nFeed      %.1f      %.1f
           %.1f   ", 0.4, 3.6, 4.0);
79 printf("\nProduct     %.1f      %.1f
           %.1f   ", 0.4, 0.4, 0.8);
80 printf("\nEvaporation  %.1f      %.1f
           %.1f   ", 3.6-0.4, 4.0-0.8);
81
82 // Assuming , as an approximation , equal evaporation
   in each effect , or D1 = D2 = D3 = 1.07 kg/s , then
   the latent heat of flash vaporisation in the
   second effect is given by:
83 q = 4.18*(4-1.07)*(376-358);
84 printf("\n latent heat of flash vaporisation in the
   second effect is = %.1f kW",q);
85 printf("\n latent heat of flash vaporisation in the
   third effect is := %.1f kW ",4.18*(4-2*1.07)
   *(358-325));
86 printf("\n At 394 K, the latent heat = 2200 kJ/kg");
87 printf("\n At 325 K, the latent heat = 2377 kJ/kg");
88 printf("\n Mean value ,      = 2289 kJ/kg");
89
90 printf("\n Values of T1 ,T2,T3 are now chosen by trial
   and error to give equal values of A in each
   effect , as follows");
91 printf("\n deltaT1      A1      deltaT2      A2
           deltaT3      A3");
92 printf("\n ( deg K )      (m^2)      ( deg K )      (m^2)      ( deg K
           )      m^2");
93 printf("\n      18          64.2      18          61.4      33
           66.9 ");
94 printf("\n      19          60.5      17          65.0      33
           66.9 ");
95 printf("\n      18          64.2      17.5        63.1      33.5
           65.9 ");
96 printf("\n      18          64.2      17          65.0      34
           64.9 ");
97

```

```

98 printf("\n Steam consumption = %.2f kg/s" ,3580/2289)
      ;
99 printf("\n Economy = %.1f kg/kg" ,3.2/1.56);

```

---

### Scilab code Exa 14.2B Backward feed

```

1 clear;
2 clc;
3 printf("\n Example 14.2b");
4 deltaT1 = 20;           // temperature is in deg K
5 deltaT2 = 24;           // temperature is in deg K
6 deltaT3 = 25;           // temperature is in deg K
7 To = 394;               // temperature is in deg K
8 T1 = 374;               // temperature is in deg K
9 T2 = 350;               // temperature is in deg K
10 T3 = 325;              // temperature is in deg K
11 latent_heat0 = 2200;    // latent heat in kJ/kg
12 latent_heat1 = 2254;    // latent heat in kJ/kg
13 latent_heat2 = 2314;    // latent heat in kJ/kg
14 latent_heat3 = 2377;    // latent heat in kJ/kg
15
16 // Effect 3:
17 //          2314 D2 = 4     4.18(325)     294) +
18     2377 D3
19
20 // Effect 2:
21 //          2254 D1 = (4     D3) 4.18(350)     325)
22     + 2314 D2
23
24 // Effect 1:
25 //          2200 D0 = (4     D3     D2) 4.18(374
26     350) + 2254 D1
27
28 // (D1 + D2 + D3) = 3.2 kg/s

```

```

27 D = [0 2314 -2377;2254 -2314 4.18*(350-325);1 1 1];
28 C = [4*4.18*(325-294);4*4.18*(350-325);3.2];
29 D1 = inv(D)*C;
30 printf("\n D1 = %.3f kg/s",D(1));
31 printf("\n D2 = %.3f kg/s",D(2));
32 printf("\n D3 = %.3f kg/s",D(3));
33 A1 = 1.387*latent_heat0/(2.5*20);
34 A2 = 1.261*latent_heat1/(2*24);
35 A3 = 1.086*latent_heat2/(1.6*25);
36 printf("\n A1 = %.1f m^2",A1);
37 printf("\n A2 = %.1f m^2",A2);
38 printf("\n A3 = %.1f m^2",A3);

```

---

### Scilab code Exa 14.3 Efficiency of the compressor

```

1 clear;
2 clc;
3 printf ('\n Example 14.3');
4 He = 2690;           //He is the enthalpy of entrained
                      steam in kJ/kg
5 H4 = ((1*2780)+(1.6*2690))/2.60;
6
7 //Again assuming isentropic compression from 101.3
   to 135 kN/m2, then:
8 H3 = 2640;           //in kJ/kg (from the chart)
9 n = (1.0+1.6)*(2725-2640)/[1.0*(2780-2375)];
10 printf ("\n      = %.2f ",n);
11 printf ("\n This value is low, since in good design
          overall efficiencies approach 0.75 0 .80.
          Obviously the higher the efficiency the greater
          the entrainment ratio or the higher the saving in
          live steam");
12 Pe = 101.3;          //pressure of entrained vapour in
                      kN/m2
13 discharge_P = 135;// discharge pressure in kN/m2

```

```

14 printf("\n the required flow of live steam = 0.5 kg/
    kg entrained vapour.");
15 printf("\n In this case the ratio is (1.0/1.6) =
    0.63 kg/kg");

```

---

### Scilab code Exa 14.4 Quantity of additional stream required

```

1 clear;
2 clc;
3 printf("\n Example 14.4");
4 //Making a mass balance , there are two inlet streams
    -
5 //the additional steam , say Gx kg/s
6 //the sea water feed , say Gy kg/s .
7
8 //The two outlet streams are
9 //distilled water product , 0.125 kg/s
10 //the concentrated sea water , 0.5 Gy kg/s
11
12 //Thus: (Gx + Gy) = (0.125 + 0.5 Gy) or (Gx + 0.5 Gy
    ) = 0.125
13
14 //At 650 kN/m2, the total enthalpy of the steam =
    2761 kJ/kg .
15
16 //Thus the energy in this stream = 2761Gx kW
17 //sea water enters at 344 K
18 //enthalpy of feed = [Gy      4.18(344      273)] =
    296.8Gy kW
19 printf("\n The enthalpy of the product is 439*0.125
    = 54.9 kW");
20
21 //Making a balance
22 //(E + 2761Gx + 296.8Gy) = (211.3Gy + 54.9)
23 //(E + 2761Gx + 85.5Gy) = 54.9

```

```

24 //where E is the power supplied to the compressor
25 // $(E + 2590G_x) = 33.5$ 
26
27 //For a single-stage isentropic compression
28 //the work done in compressing a volume V1 of gas at
    pressure P1 to a volume V2 at pressure P2 is
        given by
29 P1 = 101.3;           //pressure is in kN/m^2
30 P2 = 120;             //pressure is in kN/m^2
31 dsteam_P1 = (18/22.4)*(273/374.1); //density of
    steam at 101.3 kN/m^2 and 374.1 K
32 // $V_1 = 0.5G_y/dsteam\_P1 = 0.853 \text{ Gy m}^3/\text{sec}$ 
33 E = poly([0], 'E');
34 E1 = roots(0.5*E-(P1*0.853)/(1.3-1)*[(P2/P1)
    ^((0.3/1.3)-1)]);
35 printf("\n E = %.1f kW/(kg/sec)", E1);
36
37 printf("\n As the compressor is 50 per cent
    efficient then  $E = %.1f \text{ kW/(kg/sec)}$ ", E1/0.5);
38 // $E = 46 * 0.5 * \text{Gy} = 23.0 \text{ Gy kW}$ 
39 Gx = poly([0], 'Gx');
40 Gx1 = roots(2761*Gx - 54.9 - 4.72*(33.5-2590*Gx));
41 printf("\n Gx = %.3f kg/sec", Gx1);

```

---

### Scilab code Exa 14.5 Capacity and economy of the system

```

1 clear;
2 clc;
3 printf("\n Example 14.5");
4 //Properties of the ejector
5 e1 = 0.95;           //nozzle efficiency
6 e2 = 0.80;           //efficiency of momentum
    transfer
7 e3 = 0.90;           //efficiency of compression
8 P1 = 650;            //pressure of live stream

```

```

9 P2 = 101.3;           //pressure of entrained steam
10 H1 = 2970; //The enthalpy of the live steam at 650 kN
    /m2 and (435 + 100) in kJ/kg
11 H2 = 2605; //the enthalpy after isentropic expansion
    from 650 to 101.3 kN/m2, using an
    enthalpy entropy chart, in kJ/kg
12 x2 = 0.97;          //dryness fraction
13
14 printf("\n The enthalpy of the steam after actual
    expansion to 101.3 kN/m2 is %d kJ/kg", e1*(H1 -
    H2));
15 LatentHeat = 2258;    //latent heat at 101.3 kN/m
    ^2
16 //dryness after expansion but before entrainment x21
    is given by:
17 //((x21 - x2) = (1 - e1)(H1 - H2)
18 x21 = poly([0], 'x21');
19 x22 = roots((x21-x2)*LatentHeat-(1-e1)*(H1-H2));
20 printf("\n dryness after expansion but before
    entrainment x21 is = %.3f", x22);
21
22 //If x23 is the dryness after expansion and
    entrainment, then:
23 //((x23 - x2) = (1 - e3)(H1 - H2)
24 x23 = poly([0], 'x23');
25 x24 = roots((x23-x2)*LatentHeat-(1-e3)*(H1-H2));
26 printf("\n the dryness after expansion and
    entrainment is %.2f", x24);
27
28 P3 = 205;            //discharge pressure in kN/m2
29 //Assuming that the steam at the discharge pressure
    P3 = 205 kN/m2 is also saturated
30 x3 = 1;
31 H3 = 2675; //enthalpy of the mixture at the start of
    compression in the diffuser section at 101.3 kN/
    m2
32
33 //Again assuming the entrained steam is also

```

saturated , the enthalpy of the mixture after isentropic compression in the diffuser from 101.3 to 205 kN/m<sup>2</sup> is H<sub>4</sub>

```

34 H4 = 2810;           // in kJ/kg
35 m = {[ (H1-H2)/(H4-H3)]*e1*e2*e3-1}; //m = m2/m1
36 printf("\n m2/m1 = %.2f kg vapour entrained/kg kive
            stream",m);
37 printf("\n Thus with a flow of 0.14 kg/s live steam ,
            the vapour entrained at 101.3 kN/m2 is %.2f kg/
            sec",0.14*m);
38 printf ('\n Total saturated steam = %.2f kg/sec at
            205 kN/m^2',0.14+0.14*m);
39 printf ("\n The economy of the steam = %.2f
            ",0.214/0.14);
40 printf ("\n The capacity in terms of throughput of
            solution Gf = %.3f kg/sec",0.214+0.025);

```

---

### Scilab code Exa 14.6 Method to drive the compressor

```

1 clear;
2 clc;
3 printf("\n Example 14.6");
4 //(a) Diesel engine
5 printf("\n (a) Diesel engine");
6
7 printf("\n For 1 kg evaporation , ammonia circulated
            = 2.28 kg");
8 printf("\n Work done in compressing the ammonia = %d
            MJ/kg",150*2.28);
9 printf("\n For an output of 1 MJ, the engine
            consumes 0.4 kg fuel.");
10 printf("\n fuel consumption = %.3f kg/kg water
            evaporated",0.4*0.342);
11 printf("\n cost = %.5f euro/kg water evaporated "
            ,0.02*0.137);

```

```

12
13 printf("\n (b) Turbine");
14
15 printf("\n The work required is 0.342 MJ/kg
    evaporation");
16 //Therefore with an efficiency of 70 per cent:
17 printf("\n energy required from steam = %.3f MJ/kg "
    ,0.342*100/70);
18 printf("\n Enthalpy of steam saturated at 700 kN/m2
    = 2764 kJ/kg.");
19 printf("\n Enthalpy of steam saturated at 101.3 kN/
    m2 = 2676 kJ/kg");
20 printf("\n thus : energy from steam = %d kg/kg"
    ,2764-2676);
21 printf("\n steam required = %.2f kg/kg evaporation "
    ,0.489/0.088);
22 printf("\n Cost = %.4f euro/kg water evaporated"
    ,0.01*5.56/10);
23 printf("\n Hence the Diesel engine would be used for
    driving the compressor");

```

---

### Scilab code Exa 14.7 Optimum boiling time

```

1 clear;
2 clc;
3 printf("\n Example 14.7");
4 // (a) Maximum throughput
5
6 printf("\n (a) Maximum throughput");
7 t_bopt = (15*10^3)+(2/(7*10^(-5)))*(7*10^(-5))
    *0.2*15*10^3)^0.5;
8 printf("\n The boiling time to give maximum heat
    transfer and hence maximum throughput is %.1f
    ksecs ",t_bopt*10^(-3));
9

```

```

10 Qb = (2*40*40)*(7*10^(-5))*[((7*10^(-5)*2.81*10^4)
+0.2)^0.5-0.2^0.5];
11 printf("\n The heat transferred during boiling is %
.2f*10^7 kJ",Qb);
12 printf("\n the water vaporated = %.2f*10^4 kg"
,(4.67*10^(7)/2300)*10^(-4));
13 printf("\n Rate of evaporation during boiling = %.3f
kg/sec", (2.03*10^4)/(2.81*10^4));
14 printf("\n Mean rate of evaporation during the cycle
= %.3f kg/sec", 2.03*10^(4)/[(2.8*10^4)
+(15*10^3)]);
15 printf("\n cost = %.1f euro/cycle", ((2.81*10^(4)*18)
/1000)+600);
16
17
18 printf("\n (b) Minimum cost");
19 t_bopt1 = (600/0.018)+[2*(7*10^(-5)*0.2*600*0.018)
^0.5]/(7*10^(-5)*0.018);
20 printf("\n The boiling time to give minimum cost is
%.1f ksecs",t_bopt1*10^(-3));
21 Qb1 = [(2*40*40)/(7*10^(-5))]*[(7*10^(-5)*5.28*10^4
+ 0.2)^0.5-0.2^0.5];
22 printf("\n The heat transferred during one boiling
period is %.2f*10^7 kJ",Qb1*10^(-7));
23 printf("\n The water evaporated = %.2f*10^4 kg",Qb1
/2300);
24 printf("\n Rate of evaporation = %.3f kg/sec"
,(3.03/5.28));
25 printf("\n Mean rate of evaporation during the cycle
= %.2f kg/sec", (3.03*10^4)/[(5.28*10^4)
+(15*10^3)]);
26 printf("\n cost of one cyce = %.1f euro", 5.28*10^(4)
*0.018+600);

```

---

# Chapter 15

## Crystallisation

Scilab code Exa 15.1 Supersaturation ratio

```
1 clear;
2 clc;
3 printf("\n Example 15.1");
4 printf("\n For concentrations in kg sucrose/kg water :");
5 c = 2.45; //concentration is in kg/kg
6 printf("\n c = %.2f kg/kg",c);
7 c1= 2.04; //concentration is in kg/kg
8 printf("\n c1 = %.2f kg/kg",c1);
9 S = c/c1;
10 printf("\n S = %.2f",S);
11
12 printf("\n\n For concentrations in kg sucrose/kg
water:")
13 co = c/(c+1); //concentration is in kg/kg
solution
14 printf("\n co = %.3f kg/kg solution",co);
15 co1 = c1/(c1 + 1); //concentration is in kg/kg
solution
16 printf("\n co1 = %.3f kg/kg solution",co1);
17 S = co/co1;
```

```
18 printf("\n S = %.2f ",S);
```

---

### Scilab code Exa 15.2 Increase in solubility

```
1 clear;
2 clc;
3 printf("\n Example 15.2");
4 printf("\n For barium sulphate");
5
6 function [x] = increase_barium(d)
7     y = exp((2*233*0.13)/(2*8314*298*4500*d));
8     x = y-1;
9     funcprot(0);
10 endfunction
11 i=1;
12 d = [0.5 0.05 0.005];
13 while(i<=3)
14     printf("\n Increase in solubility for barium
15         sulphate fpr particle size %f um = %f per
16         cent",d(i)*2,(increase_barium(d(i)*10^(-6))
17         *100));
18     i = i+1;
19 end
20
21 printf("\n For Sucrose");
22 function [a] = increase_sucrose(d1)
23     b = exp((2*342*0.01)/(1*8314*298*1590*d1));
24     a = b-1;
25     funcprot(0);
26 endfunction
27 j=1;
28 while(j<=3)
29     printf("\n Increase in solubility for barium
30         sulphate fpr particle size %f um = %f per
31         cent",d(j)*2,(increase_sucrose(d(j)*10^(-6)))
```

```
    *100));  
27     j = j+1;  
28 end
```

---

### Scilab code Exa 15.3 Theoretical yield

```
1 clear;  
2 clc;  
3 printf("\n Example 15.3");  
4 R = 322/142;  
5 //The initial concentration  
6 c1 = 1000/5000;           // Concentration is in  
                           kg Na2SO4/kg water  
7 printf("\n The initial concentration c1 = %.1f kg  
Na2SO4/kg water",c1);  
8 //The solubility  
9 c2 = 9/100;                // solubility is in Kg  
                           Na2SO4/kg water  
10 printf("\n The solubility c2 = %.2f Kg Na2SO4/kg  
water",c2);  
11 printf("\n Initial mass of water,w1 = 5000 kg and  
the water lost by evaporation E = %f kg/kg"  
,2/100);  
12 printf("\n yield y = %d kg Na2SO4.10H2O", (5000*2.27)  
*[0.2-0.09*(1-0.02)]/[1-0.09*(2.27-1)]);
```

---

### Scilab code Exa 15.4 Yield of Sodium acetate

```
1 clear;  
2 clc;  
3 printf("\n Example 15.4");  
4 //Given data:  
5 //Heat of crystallisation , q = 144 kJ/kg trihydrate
```

```

6 //Heat capacity of the solution , Cp = 3.5 kJ/kg deg
K
7 //Latent heat of water at 1.33 kN/m2, = 2.46 MJ/
kg
8 //Boiling point of water at 1.33 kN/m2 = 290.7 K
9 //Solubility of sodium acetate at 290.7 K, c2 =
0.539 kg/kg water
10
11 //Equilibrium liquor temperature
12 T = 290.7+11.5;
13 printf("\n Equilibrium liquor temperature is%.1f K",
T);
14
15 //Initial concentration
16 c1 = 40/(100-40);
17 printf("\n Initial concentration c1 = %.3f kg/kg
water",c1);
18
19 //Final concentration
20 c2 = 0.539;
21 printf("\n Final concentration ,c2 = %.3f kg/kg water
",c2);
22
23 //Ratio of molecular masses
24 printf("\n Ratio of molecular masses ,R = %.2f"
,136/82);
25
26 E = {144*1.66*(0.667-0.539)+3.5*(353-302.2)
*(1+0.667)*[1-0.539*(1.66-1)
]}/{2460*[1-0.539*(1.66-1)]-(144*1.66*0.539)};
27 printf("\n\n E = %.3f kg/kg water originally
present",E);
28 printf("\n yeild y = %.3f kg/sec", (0.56*(100-40)
/100)*1.66*[0.667-0.539*(1-0.153)
]/[1-0.539*(1.66-1)]);

```

---

### Scilab code Exa 15.5 Length of crystalliser

```
1 clear;
2 clc;
3 //The molecular mass of hydrate/molecular mass of
   anhydride
4 R = 380/164;
5 printf("\n The molecular mass of hydrate/molecular
   mass of anhydride ,R = %.2f",R);
6 //It will be assumed that the evaporation is
   negligible and that E = 0.
7 c1 = 0.23;
8 printf("\n The initial concentration , c1 = %.2f kg/
   kg solution or %.2f kg/kg water",c1,c1/(1-c1));
9 c2 = 15.5;
10 printf("\n The final concentration , c2 = %.1f kg/kg
   water or 0.155 kg/kg water",c2);
11 w1 = 0.77;
12 printf("\n In 1 kg of the initial feed solution ,
   there is 0.23 kg salt and 0.77 kg water and hence
   w1 = %.2f kg",w1);
13 y = 2.32*0.77*[0.30-0.155*(1-0)]/[1-0.155*(2.32-1)];
14 printf("\n yeild y = %.2f kg",y);
15 printf("\n In order to produce 0.063 kg/s of
   crystals , the required feed is: %.3f kg/sec"
   ,1*0.063/0.33);
16 q = 0.193*3.2*(313-298);
17 printf("\n The heat required to cool the solution %
   .1f kW",q);
18 q1 = 0.063*146.5;
19 printf("\n Heat of crystallisation = %.1f kW",q1);
20 printf("\n total heat loss = %.1f kW",q+q1);
21 printf("\n Assuming counter current flow");
22 deltaT1 = (313-293);
```

```

23 printf("\n deltaT1 = %d deg K",deltaT1);
24 deltaT2 = 298 - 288;
25 printf("\n deltaT2 = %d deg K",deltaT2);
26 deltaTlm = (deltaT1-deltaT2)/log(deltaT1/deltaT2);
27 printf("\n The logarithmic mean temperature is %.1f
deg K",deltaTlm)
28
29 A = (q+q1)/(0.14*14.4);
30 printf("\n The heat transfer area required , A= Q/U
31 deltaTm = %.1f m^2",A);
32 printf("\n Length of heat exchanger required = %.1f"
,A)
33
34 printf("\n\n\n\n In practice 3 lengths , each of 3 m
length would be specified.");

```

---

### Scilab code Exa 15.6 Crystal production rate

```

1 clear;
2 clc;
3 printf("\n Example 15.6");
4 down_time = (4/0.00017);
5 printf("\n Draw down time = %d secs",down_time);
6 printf("\n\n (a)");
7 printf("\n from a mass balance the total mass of
solids is :")
8 cs = (6*0.6*2660*10^13)*(10^(-8)*23530)^4;
9 printf("\n cs = %d kg/m^3",cs);
10 printf("\n\n (b)");
11 printf("\n The production rate = %.3f kg/sec",cs
*0.00017);
12
13 printf("\n The crystal population decreases
exponentially with size ")

```

```

14 x = exp((-100*10^(-6))/(10^(-8)*23530));
15 printf("%d percent",x*100);
16 printf("\n Thus 34 percent have been discharged by
           the time they reach 100 um");
17 printf("\n\n(c)");
18 // If (100 - 90) = 10 per cent of the nuclei remain
       and grow to >100 m , then
19 t = poly([0], 't');
20 t1 = roots(t - 10^(4)/log(1/0.10));
21 printf("\n tr = %d secs",t1);
22 Q = poly([0], 'Q');
23 Q1 = roots(Q + (4/4343-0.00017));
24 printf("\n Qf = %f m^3/sec", -Q1);

```

---

### Scilab code Exa 15.7 Vapour pressure

```

1 clear;
2 clc;
3 printf("\n Example 15.7");
4 l = poly([0], 'l');
5 l1 = roots(log(780/220)-[l*(463-433)
                           ]/[8314*463*433]);
6 printf("\n v = %d kj/kmol",l1);
7 P = poly([0], 'P');
8 P1 = roots(P-220/exp(70340*(433-393)/(8.314*433*393)
                      ));
9 printf("\n P = %d kN/m^2",P1);

```

---

### Scilab code Exa 15.8 Mass sublimation rates

```

1 clear;
2 clc;
3 printf("\n Example 15.8");

```

```
4 printf("\n Vaporisation stage :");
5 Pt = 101.5;
6 Ps = 1.44;
7 Sv = 0.56*(138/29)*(1.44/(101.5-1.44));
8 printf("\n Sv = %.3f kg/sec",Sv);
9
10 printf("\n\n COndensation stage: ")
11 Ptc = 100;
12 Psc = 0.0023;
13 S = 0.56*(138/29)*(0.0023/(100-0.0023));
14 printf("\n S = %f kg/sec",S);
```

---

# Chapter 16

## Drying

**Scilab code Exa 16.1** Time taken to dry the solids

```
1 clear;
2 clc;
3 printf("\n Example 16.1");
4 //For the first drying operation
5 w1 = 0.25;                      //in kg/kg
6 w = 0.10;                        //in kg/kg
7 wc = 0.15;                       //in kg/kg
8 we = 0.05;                        //in kg/kg
9 f1 = w1-we;                      //in kg/kg
10 fc = wc - we;                   //in kg/kg
11 f = w - we;                     //in kg/kg
12
13 //x = mA
14 function[x]=TotalDryingTime(t)
15     x = (1/t)*[(f1-fc)/fc + log(fc/f)];
16     funcprot(0);
17 endfunction
18 printf("\n      mA = %.3f kg/sec",TotalDryingTime(15))
19
20 //For the second drying operation
```

```

21 w12 = 0.30; //in kg/kg
22 w2 = 0.08; //in kg/kg
23 wc2 = 0.15; //in kg/kg
24 we2 = 0.05; //in kg/kg
25 f12 = w12 - we2; //in kg/kg
26 fc2 = wc2 - we2; //in kg/kg
27 f2 = w2 - we2; //in kg/kg
28
29 t1=(1/TotalDryingTime(15))*(0.25-0.10)/0.10 + log
   (0.10/0.03)];
30 printf("\n The total drying time is then %.1f ksec
   or %.2f hr ",t1,t1*1000/3600);

```

---

### Scilab code Exa 16.2 Time taken to dry the solids

```

1 clear;
2 clc;
3 printf("\n Example 16.2");
4 E = [1 0.64 0.49 0.38 0.295 0.22 0.14];
5 J = [0 0.1 0.2 0.3 0.5 0.6 0.7];
6 plot(J,E,rect=[0,1,0,1]);
7 xtitle("Plot for drying data","J = kt/L^2","E");
8
9 //For the 10 mm strips
10 mi = (0.28 - 0.07); // Initial free moisture
   content in kg/kg
11 mf = (0.13-0.07); //Final free moisture
   content in kg/kg
12 //at
13 t = 25; //time is in ksecs
14 E = (0.06/0.21);
15 //at E = 0.286 ,J = 0.52 from the plot of given data
   and J = kt/L^2
16 k = poly([0], 'k');
17 k1 = roots(0.52 - (k*t)/(10/2)^2);

```

```

18 printf("\n k = %f",k1);
19
20 //for the 60 mm planks
21 m1i = (0.22 - 0.07);           //Initial free moisture
22 m1f = (0.13 - 0.07);           //Final free moisture
23 E = (m1f/m1i);
24 //at E = 0.20 from yhe plot of the given data J =
25          0.63 and J = kt/L^2
25 t1 = 0.63*(60/2)^(2)/k1;
26 printf("\n t1 = %d ksecs or %.1f days",t1,(t1
27          *1000/(3600*24)));

```

---

### Scilab code Exa 16.3 Mass flow rate of dry air

```

1 clear;
2 clc;
3 printf("\n example 16.3");
4 //(a) Air
5 //G kg/s dry air enter with 0.006G kg/s water vapour
      and hence the heat content of this stream=
6 //[(1.00G) + (0.006G    2.01)](385      273) = 113.35
      G kW
7
8 //(b) Wet solid
9 //0.125 kg/s enter containing 0.40 kg water/kg wet
      solid, assuming the moisture is expressed on a
      wet basis.
10 flowWater = (0.125*0.40);           //in kg/sec
11 flowDrySolid = (0.125-0.050);       //in kg/sec
12 //Hence heat content of this stream
13 q = [(0.050*4.18)+(0.075*0.88)]*(295-273);
14 printf("\n The heat content of this stream = %.2f kW
      ",q);

```

```

15
16 //Heat out
17 //(a) Air
18 //Heat in exit air = [(1.00 G) + (0.006 G      2.01)
19 //                         ](310      273) = 37.45G kW.
20 fd = 0.075;           //mass flow rate of dry
21 //solids in kg/sec
22 w = 0.05*0.075/(1+0.05); //water in the dried
23 //solids leaving in kg/secs
24 we = (0.050 - w);       //The water evaporated
25 //into gas stream in kg/secs
26
27 //Assuming evaporation takes place at 295 K, then:
28 qout = 0.0464*[2.01*(310-295)+2449+4.18*(295-273)];
29 printf("\n Heat in the water vapour = %.1f kW",qout)
;
30
31 //the total heat in this stream = (119.30 + 37.45G)
32 //kW.
33 //(b) Dried solids
34
35
36 //The dried solids contain 0.0036 kg/s water and
37 //hence heat content of this stream is:
38 q1 = [(0.075*0.88)+(0.0036*4.18)/(305-273)];
39 printf("\n The dried solids contain 0.0036 kg/s
40 //water and hence heat content of this stream is
41 // : %.2f kW",q1);
42
43 //((c) Losses
44 //These amount to 20 kJ/kg dry air or 20m kW.
45 //Heat Balance
46 G = poly([0], 'G');
47 G1 = roots(113.35*G + 6.05 - 119.30 - 37.45*G - 2.59
48 //           - 20*G);
49 printf("\n G = %.2f kg/secs",G1);
50 printf("\n Water in the outlet stream %.4f kg/secs"

```

```

        ,0.006*2.07+0.0464);
43 printf("\n The humidity H = %.4f kg/kg dry air"
        ,0.0588/2.07);

```

---

### Scilab code Exa 16.4 Approximate drying time

```

1 clear;
2 clc;
3 printf("\n Example 16.4");
4 //In 100 kg of feed
5
6 //mass of water =
7 mw = 100*30/100;                                //mass of water
     in kg
8 //mass of dry solids =
9 md = 100-30;                                     //mass of dry
     solids
10
11 //and:
12 //For b kg water in the dried solids: 100b/(b + 70)
13 b = poly([0], 'b');
14 b1 = roots(100*b - 15.5*(b+70));
15 printf("\n water in the product ,b = %.1f kg",b1);
16
17 //Initial water content
18 w1 = 30/70;                                       //Intial moisture content
     in kg/kg dry solids
19 //Final moisture content
20 w2 = (12.8/70);                                  //Final moisture content
     in kg/kg dry solids
21 //water to be removed
22 w3 = (30-12.8);                                 //water to be removed in
     kg
23

```

```

24 // Surface of drying
25 S = (0.03*70);           // Surface for drying in m
                           ^2
26 rate = (0.0007*2.1);    //Rate of drying in kg/sec
27
28 //As the final moisture content is above the
   critical value , all the drying is at this
   constant rate and the time of drying is:
29 t = 17.2/0.00147;
30 printf("\n As the final moisture content is above
   the critical value , all the drying is at this
   constant rate and the time of drying is: %d ksecs
   or %.2f hr",t,t/3600);

```

---

### Scilab code Exa 16.5 Proposed diameter and length

```

1 clear;
2 clc;
3 printf("\n Example 16.5");
4 //Inlet air temperature
5 Tair = 400;           //Inlet air temperature in
                        kelvins
6 H = 0.01;             //Humidity is in kg/kg dry
                        air
7 /*therefore wet bulb temperature =
8 Twetbulb = 312;       //Inlet wet-bulb temperature
9 NTU = 1.5;             //Number of transfer units
10
11 //Then for adiabatic drying the outlet air
   temperature ,To is given by
12 To = poly([0], 'To');
13 To1 = roots(exp(1.5)*(To-312)-(400-312));
14 printf("\n For adiabatic drying the outlet air
   temperature = %.1f K",To1);
15

```

```

16 // Solids outlet temperature will be taken to be
   maximum allowable ,325K
17 //From the steam tables in the Appendix , the latent
   heat of vaporisation of water at 312 K is 2410 kJ/
   kg. Again from steam tables , the specific heat
   capacity of water vapour = 1.88 kJ/kg K and that
   of the solids will be taken as 2.18 kJ/kg K.
18
19 //For a mass flow of solids of 0.35 kg/s and inlet
   and outlet moisture contents of 0.15 and 0.005 kg
   /kg dry solids respectively , the mass of water
   evaporated = 0.35(0.15 - 0.005) = 0.0508 kg/s.
20
21 //For unit mass of solids ,the heat duty includes:
22 //Heat to the solids
23 qsolids = 2.18*(325-300);           //heat to
   solids in kJ/kg
24 //Heat to raise the moisture to the dew point
25 qdew = (0.15*4.187*(312-300));    //in kJ/kg
26 //Heat of vaporisation
27 qvap = 2410*(0.15-0.005);         //in kJ/kg
28 //Heat to raise remaining moisture to solids
   outlet temperature
29 qremaining = (0.05*4.187)*(325-312);
30 //Heat to raise evaporated moisture to the
   air outlet temp.
31 qevapo = (0.15-0.005)*1.88*(331.5-312);
32 qtotal = qsolids + qdew + qvap + qremaining +
   qevapo;
33 printf("\n Total heat = %.1f kJ/kg or %d kW",
   qtotal,qtotal*0.35);
34
35 //The humid heat of entering air is 1.03 kJ/kg K
36 //G1 (1 +H1) = Q/Cp1(T1 - T2)
37 //where: G1 (kg/s) is the mass flowrate of
   inlet air ,
38 //H1 (kg/kg) is the humidity of inlet air ,
39 //Q (kW) is the heat duty ,

```

```

40 //Cp1 (kJ/kg K) is the humid heat of inlet
41 //air
42 //and: T1 and T2 (K) are the inlet and outlet
43 //air temperatures respectively .
44 G1 = poly([0], 'G1');
45 G = roots(G1*(1+0.01)-146/(1.03*(400-331.5)))
46 ;
47 printf("\n Mass flow rate of inlet air = %.2f
48 kg/secs", G);
49 printf("\n Mass flow rate of dry air ,Ga = %
50 .2f kg/secs", G/1.01);
51 printf("\n the humidity of the outlet air H2
52 = %.4f kg/kg" ,0.01+0.0508/2.05);
53
54 //At a dry bulb temperature of 331.5 K, with a
55 //humidity of 0.0347 kg/kg, the wet-bulb
56 //temperature of the outlet air, from Figure 13.4 in
57 //Volume 1, is 312 K, the same as the inlet, which
58 //is the case for adiabatic drying.
59
60 //The dryer diameter is then found from the
61 //allowable mass velocity of the air and the
62 //entering air flow and for a mass velocity of 0.95
63 //kg/m^2.secs , the cross sectional area of the
64 //dryer is
65 printf("\n The cross sectional area of the dryer is
66 %.2f m^2" ,2.07/0.95);
67 printf("\n The equivalent diameter of the dryer = %
68 .2f m" ,[(4*2.18)/%pi]^0.5);
69
70 //With a constant drying temperature of 312 K:
71 //at the inlet
72 deltaT1 = 400-312; //inlet
73 //temperature is in deg K
74 //at the outlet
75 deltaT2 = 331.5-312; //outlet
76 //temperature is in deg K
77 Tlogmean = (deltaT1 - deltaT2)/log(deltaT1/

```

```

        deltaT2);
60      printf("\n Logarithmic mean temperature
               difference = %.1f deg K", Tlogmean);
61 //The length of the dryer , L is then: L = Q/(0.0625
       DG^(0.67)*Tm)
62 //where D (m) is the diameter
63 //and G(kg/m^2. secs) is the air mass velocity .
64 L = 146/[0.0625*(%pi)*1.67*(0.95)^(0.67)*45.5];
65 printf("\n The length of the dryer = %.1f m",L);
66 printf("\n Length/diameter ratio = %d "
           ,10.1/1.67);

```

---

### Scilab code Exa 16.6 Specified diameter of the bed

```

1 clear;
2 clc;
3 printf("\n Example 16.6");
4 H = 0.036;                                //Humidity is in kg/kg at
811 K
5 //Taking R as 90 per cent and P as 101.3 kN/m2, then
   , for assumed values of Tb of 321, 333 and 344 K
6 //Pw = 13,20 and 32 kN/m2, respectively
7 //G = 27.8, 12.9 and 6.02 kg/s, respectively .
8
9 //for Tb = 321, 333 and 344 K,
10 //G = 7.16, 7.8 and 7.54 kg/s respectively .
11 Tb = [321 333 344];
12 G1 = [27.8 12.9 6.02];                  //Temperature is in
   kelvins
13 G = [7.16 7.8 7.54];                   // flow rate in kg/
   secs
14 plot2d(Tb,G,style=3);
15 plot2d(Tb,G1,style=2);
16 xtitle("Temperature vs Flow rate","Temperature Tb(K)
           ","Flow rate G(kg/secs)");

```

```
17
18
19 // Plotting G against Tb for each equation on the
   same axis , then
20 Go = 8.3;                                //Gas flow rate is
   in kg/secs
21 Tb = 340;                                 //temperature is
   in Kelvins
22 uf = 0.61;                               // velocity is in m
   /secs
23
24 D = sqrt(340*(8.3+(1.58*1.26))/(278*0.61));
25 printf("\n D = %.2f m",D);
```

---

# Chapter 17

## Adsorption

**Scilab code Exa 17.1** Comparison of estimates with the geometric surfaces

```
1 //Example 17.1
2 clear;
3 clc;
4 printf("\n Example 17.1");
5 //For 1 m3 of pellet with a voidage , then
6 //Number of particles = (1      )/(    /6)(15
7 //          10-9)3
8 //Surface area per unit volume = (1      )
9 //          (15      10-9)2/(    /6)(15      10-9)3
10 //= 6(1      )/(15      10-9) m2/m3
11 //1 m3 of pellet contains 2290(1-      )kg
12 //solid and hence:
13 specific_surface = 6/(15*10^(-9)*2290);
14 printf("\n Specific surface = %.3f*10^5 m^2/
15 kg ",specific_surface*10^(-5));
16 // (a) Using the BET isotherm
```

```

17      y1 = [1666 3333 5109 7138 9226 11358];
18      //x = P/Po
19      x = [0.1 0.2 0.3 0.4 0.5 0.6];
20      plot2d(x,y,style = 2);
21      xtitle("","P/Po","(P/Po)/(V *10^(-6)) or (P/
Po)/(V *(1-(P/Po)))");
22      plot2d(x,y1,style=3);
23      legend(" Langmuir isotherm ","(BET) isotherm")
;
24
25      //intercept , 1/VB = 300, and slope , (B - 1)
//VB = 13,902
26      B = (13902/300)+1;
27      printf("\n B = %.2f ",B);
28      V = 1/(300*47.34);
29      //Total surface area
30      S = [(70.4*10^(-6)*808*6.2*10^(26)
*0.162*10^(-18))/28;
31      printf("\n Total surface area = %.3f *10^5 m
^2/kg",S*10^(-5));
32
33 // (b) Using the Langmuir form of the isotherm
34
35 //Slope 1/V = 13,902
36 V = 1/13902;
37 printf("\n V = %.1f *10^(-6)m^3/kg",V*10^6);
38 //which agrees with the value of the BET
isotherm
39 //It may be noted that areas calculated from the
isotherm are some 20 per cent greater than the
geometric surface , probably due to the
existence of some internal surface within the
particles

```

---

**Scilab code Exa 17.2** Applicability of various equilibrium theories

```

1 //Example 17.2
2 clear;
3 clc;
4 printf("\n Example 17.2");
5 // (a)
6 //For n = 1,
7 // (P/P0)/V= (P/P0)V1+ 1/B2V1
8 //where V is the gas phase volume equivalent to
   the amount adsorbed
9 //x = (P/Po)
10 x = [0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40];
11 //y = (P/Po)/V in kg/m^3
12 y = [0.76 1.35 1.85 2.27 2.66 2.94 3.21 3.42];
13 //y1 = (P/Po)/(V*(1-P/Po)) in kg/m^3
14 y1 = [0.80 1.50 2.18 2.88 3.55 4.20 4.94 5.73];
15 plot(x,y,"o-");
16 plot(x,y1,"+-");
17 xtitle("","(P/Po)","(P/Po)/V or (P/Po)/(V*(1-P/
   Po))");
18 legend("B.E.T.","Langmuir");
19
20 //The data , which are plotted as (P /P 0)/V
   against P/P0 may be seen to conform to a
   straight line only at low values of P/P0,
   suggestin g that more than one layer of
   molecules isadsorbed .
21 Slope = 12.56;
22 V1 = 1/12.56;
23 //The surface area occupied by this absorbed
   volume
24 S = V1*6.02*10^(26)*0.162*10^(-18)/24;
25 printf("\n S = %d m^2/kg",S);
26
27
28 // (b)
29 //P/P0/V*(1      P/P0)= 1/V1*B2+( B 2 - 1 )/(V1*B2)*(P
   /P 0)
30 //y2 = (P/Po)

```

```

31 //x2 = 1V^2
32 y2 = [0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40 0.50
       0.60 0.87 0.80];
33 x2 = [230 183 152 129 113 96 84 73 53 37 26 20];
34 xset('window',1);
35 plot(x2,y2,"o-");
36 xtitle("Harkins-Jura Plot","1/V^2(kg^2/m^6)","P/Po"
         );

```

---

### Scilab code Exa 17.3 Length of the bed

```

1 //Example 17.3
2 clear;
3 clc;
4 printf("\n Example 17.3");
5 //For case (a):
6 //Cs = 10C which represents a linear isotherm.
7 //All concentrations move at the same velocity.If
     z0=0 at t = 0 for all concentrations , the
     adsorption wave propagates as a step change
     from the inlet to the outlet concentration .
8 //f (Cs ) = 10
9 //u = 1 10 ^(-4)/[( /4)*(0.15^(2)* )] m/s
10 //where      is the intergranular voidage = 0.4
11 e = 0.4;
12 m = e/(1-e);
13 printf("\n m=%f",m);
14 t = 3600;           //time is in secs
15 z = [(4*10^(-4))/(%pi*(0.15^2)*0.4)
       ]*[3600/(1+10*(0.6/0.4))];
16 printf("\n z = %.2 f m",z);
17
18
19 //It may be noted that , when the adsorption wave
   begins to emerge from the bed , the bed is

```

```

        saturated in equilibrium with the inlet
        concentration .
20 //Hence: uA tC0 = zA[ C0 + (1      )Cs ]
21
22 //For case (b):
23 //Cs = 3C0.3 which represents a favourable isotherm.
24 //As C increases , f(C) decreases and points of
        higher concentrations are predicted to move a
        greater distance in a given time than lower
        concentrations. It is not possible for points of
        higher concentrationsto overtake lower
        concentrations , and if z0=0 for all
        concentrations ,the adsorption wave will propagate
        as a step change similar to case a.
25
26 //Hence: z = ut/[1+(1/m)(Cs/Co)]
27 Co = 0.003;                      //in kmol/m^3
28 z = [(4*10^(-4)*5/((%pi)*0.4*0.15^2))
        ]*[3600/(1+(0.6/0.4)*(3/Co^0.7))];
29 printf("\n z = %.2f m",z);
30
31 //For case (c):
32 //Cs = (10^(4))*(C^2) which represents an
        unfavourable isotherm .
33 //f(C)=2* 10^4* C.
34 //As C increases , f (C) increases such that , in a
        given time , z for lower concentrations is greater
        than for higher concentrations . Following the
        progress of the breakpoint concentration ,C =
        0.003 kmol/m^3 ,then :
35 // f(0.003) = 60
36 z1 = [4*10^(-4)/(%pi*0.15^(2)*0.4)
        ]*[3600/(1+(0.60/0.40)*60)];
37 printf("\n For case (c)");
38 printf("\n z = %.2f m",z1);
39
40
41 //At breakpoint ,the bed is far from saturated and:

```

```

42 saturation = 100*[(4*10^(-4)*3600)/(%pi*(0.15^(2)
    *0.4))]/[0.55*(1+(0.6/0.4)*(9/0.03))];
43 printf("\n saturation = %.1f per cent",saturation)
    ;

```

---

### Scilab code Exa 17.5 Moving bed adsorption design

```

1 clear;
2 clc;
3 printf("\n Example 17.5");
4 //let x = [ f(C) ] mean
5 x = (8.4*10^(-2)*1266)/(2.67*10^(-3)*1.186);
6
7 //e is porosity
8 e = 0.47;
9 m = e/(1-e);
10 printf("\n m = %.2f",m);
11
12 //The velocity uc with which the adsorption wave
   moves through the column may be obtained from
   equation 17.79
13 uc = 6.2*10^(-6);
14 density = 1266;           //density is in kg/m
                           ^3
15 Gs = uc*(1-e)*density;
16 printf("\n Gs = %.2f*10^(-3) kg/m^2.sec ",Gs*10^3);
17
18
19 //From overall balance:
20 //We have given eq: Gs1(0.084 - 0) = 0.129(0.00267)
21 //On solving above eq :
22 Gs1 = poly([0], 'Gs1');
23 Gs2 = roots(Gs1*(0.084-0) - 0.129*(0.00267));
24 printf("\n Gs1 = %.2f*10^(-3) kg/m^2.sec ",Gs2
    *1000);

```

```

25
26 //Part(b)
27 //time ta for the adsorption zone to move its own
   length za is given by:
28 za = 0.185;
29 ta =(za/uc)/3600;
30 printf("\n ta = %.1f hrs",ta);
31
32 //The time taken for a point at a distance z into
   the zone to emerge is given by:
33 //t = (z1/za)ta
34 yr = [0.0001 0.0002 0.0006 0.0010 0.0014 0.0018
        0.0022 0.0024];
35 yre = [0.00005 0.0001 0.00032 0.00062 0.00100
        0.00133 0.00204 0.00230];
36 I=0;
37 I1 = 0;
38 i=1;
39 printf("\nI= %.1f",I);
40 while i <=8
41     y(i)=1/(yr(i)-yre(i));
42     if i>1 then
43         I = I + (yr(i)-yr(i-1))*(y(i)+y(i-1))/2;
44         printf("\n I=% .1f",I);
45     end
46     i=i+1;
47 end
48 z_za = [0 0.137 0.362 0.473 0.560 0.674 0.852
        1.000];
49 t = (z_za)*ta;
50 j=1;
51 while j<=8
52     printf("\n time = %.1f h",t(j));
53     j=j+1;
54 end

```

---

# Chapter 18

## Ion Exchange

**Scilab code Exa 18.1** Prediction of time t against xs values

```
1 clear;
2 clc;
3 printf("\n Example 18.1");
4 Cse=0.132;           //in kg/kg
5 Cs = [0.091 0.097 0.105 0.113 0.125 0.128 0.132];
6 C = Cs/Cse;
7 C1 = 1-C;
8 C2 = 1-C^2;
9 t = [2 4 10 20 40 60 120];
10 //xset('window',1);
11 //
12 plot(t,C1,t,C2);
13 xtitle('1-(Cs/Cs*) vs t (min)', "1-(Cs/Cs*)", 't (min)');
14 legend("1-(Cs/Cs*)", "1-(Cs/Cs*)^2");
15
16 //From the plot ^2Dr/ri^2 = 0.043
17 //For a pellet of twice the radius, that is r = 2ri
18 Slope = -0.043/4;
19 printf("\n Slope = %.3f", Slope);
20
21 //Thus, when the radius = 2ri
```

```

22 function [x]=equation1(t)
23     x = 1-(6/(%pi)^2)*exp(-Slope*t);
24     funcprot(0);
25 endfunction
26
27 //CS/CS*=[1      exp(   DR / tri ^2) ] ^ 0.5
28 // DR / ri ^2 = 0.04
29 //For a pellet twice the size
30
31 function [x1]=equation2(t)
32     x1 = [1-exp(-0.01*t)]^0.5;
33     funcprot(0);
34 endfunction
35
36 printf("\n t (min)           Cs(kg/kg)      ");
37 printf("\n          equation(i)    ");
38 t = [4 20 60];                      //t is in min
39 i=1;
40 while i<=3
41 printf("\n %f      %f
        ",t(i),Cse*equation1(t(i)),Cse*equation2(
        t(i)));
42 i=i+1;
43 end

```

---

### Scilab code Exa 18.2 Concentration of HNO<sub>3</sub> in solution

```

1 clear;
2 clc;
3 printf("\n Example 18.2");
4 c_NaNO3 = 2;                         //per cent by mass
5 printf("\n Concentration of NaNO3 is %.3f kg/m^3",(
        c_NaNO3/85)*(103/100));
6

```

```
7 //HNO3 (Molecular weight = 63 kg/kmol)
8 //Concentration = p per cent
9 //Concentration = (10p/63)(1030/1000)= 0.163p kg/m3
10 //In the solution: xNa+ = 0.242/(0.242 + 0.163p)
11 //For univalent ion exchange
12 //yNa+/(1 - yNa+) = KNa+H + [xNa+ /(1 - xNa+)]
13
14 yNa = 0.1;
15 K_NaH = 2/1.3;
16 p = poly([0], 'p');
17 p1 = roots((0.1/0.9)*[0.163*p*(0.242+0.163*p)
    ]-1.5*[0.242*(0.242+0.163*p)]);
18 printf("\n p = %d percent", p1(1));
```

---